Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

# АТЕСТАЦІЙНА РОБОТА

## Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

«Інтелектуальна система прийняття рішень для маніпуляційного робота»
(тема)

Виконав: студент 2 курсу, гр. АУТПм-19-1
Дієсперов Анатолій Валерійович
(прізвище, ініціали)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології

освітньої програми Автоматизоване управління технологічними процесами
(код і повна назва напряму)

Тип програми освітньо-професійна
(повна назва освітньої програми)

Керівник проф. каф. КІТАМ Цимбал О. М.
(посада, прізвище, ініціали)

Допускається до захисту
зав. кафедри _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

| | |
|---|---|
| Факультет | Автоматики і комп'ютеризованих технологій |
| Кафедра | Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки |
| Рівень вищої освіти | другий (магістерський) |
| Спеціальність | 151 Автоматизація та комп'ютерно-інтегровані технології |
| Тип програми | освітньо-професійна |
| Освітня програма | Автоматизоване управління технологічними процесами |

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2020 р.

# ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Дієсперову Анатолію Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Інтелектуальна система прийняття рішень для маніпуляційного робота»
затверджена наказом по університету від 02.11.2020 р. № 1510 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 16.12.2020 р.
3. Вихідні дані до роботи Мова програмування C#, специфікація OpenGL, середовище розробки Visual Studio 2019

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз систем моделювання маніпуляційних роботів
4.3 Розробка системи прийняття рішень щодо адаптації положень маніпуляційного робота
4.4 Розробка програмного забезпечення
4.5 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 17 с. формату А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Керівник (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз літератури за темою атестаційної роботи | 04.09.2020 – 22.09.2020 | виконано |
| 2 | Аналіз методів прийняття рішень | 28.09.2020 – 09.10.2020 | виконано |
| 3 | Розробка методів пошуку маршруту | 12.10.2019 – 23.10.2019 | виконано |
| 4 | Розробка ПЗ для моделювання процесів прийняття рішень РТС | 26.10.2020 – 06.11.2020 | виконано |
| 5 | Аналіз та оцінка розроблених методів прийняття рішень | 09.11.2020 – 13.11.2020 | виконано |
| 6 | Оформлення пояснювальної записки | 16.11.2020 – 02.12.2020 | виконано |
| 7 | Подання роботи у ЕК | 16.12.2020 | виконано |
| 8 | | | |

Дата видачі завдання 02.11.2020 р.

Студент _____ Дієсперов А. В.
(підпис) ( прізвище, ініціали)

Керівник роботи _____ проф. Цимбал О. М.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка містить: 90 с., 37 рис., 12 табл., 29 джерел.

МАНІПУЛЯЦІЙНИЙ РОБОТ, МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ, МЕТОДИ ПОШУКУ ШЛЯХУ, МЕТОД НАВІГАЦІЙНИХ ГРАФІВ, КІНЕМАТИКА МАПИЛУЯТОРА, OPENGL, OPENTK

Об’єкт дослідження – маніпуляційний робот.

Предмет дослідження – підвищення ефективності функціонування інтелектуальної системи управління маніпуляційним роботом.

Мета роботи – забезпечення якості управління маніпуляційним роботом на основі розробки інтелектуальної системи підтримки прийняття рішень.

Методи дослідження – дослідження системи виконувалося методами імітаційного моделювання.

В магістерській атестаційній роботі розглянуті сучасні підходи до побудови систем управління маніпуляційними промисловими роботами.

Розроблені алгоритми, які дозволяють ефективно управляти маніпулятором в 3-х мірному просторі, на базі яких було розроблено ПЗ для моделювання управління МР.

Проведено ряд експериментів у розробленому ПЗ, дана оцінка складності окремих етапів алгоритму та оцінка швидкодії всього алгоритму пошуку оптимальних маршрутів.

# ABSTRACT

The explanatory memorandum contains: 90 pages, 37 figures, 12 tables, 29 sources.

MANIPULATION ROBOT, DECISION MAKING METHODS, METHODS OF SEARCHING FOR THE ROUTE, METHOD OF NAVIGATION GRAPHS, KINEMATICS OF THE MAPILUATOR, OPENGL, OPENTK

The object of study is a manipulation robot.

The subject of study is an intelligent control system for manipulation robot.

The subject of study is to increase the efficiency of the intelligent control system of the manipulation robot.

The purpose of the work is to ensure the quality of control of the manipulation robot based on the development of an intelligent decision support system.

Research methods – system research was performed by simulation methods.

In the master's thesis are considered modern approaches to construction of control systems of manipulative industrial robot.

Algorithms have been developed that allow to effectively control the manipulator in 3-dimensional space. Based on these algorithms, software for control simulation MR was developed.

Experiments are conducted in the developed software. The complexity of different stages of the algorithm has been estimated and the speed of the algorithm for finding the optimal route has been estimated.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ТЕРМІНІВ

БК – блок керування

КР – контролер руху

КС – координатна система

МР – маніпуляційний робот

НГ – навігаійний граф

ОЗК – обернена задача кінематики

ПЗ – програмне забезпечення

ПК – промисловий комп'ютер

ПТ – планувальник траєкторії

РТК – робототехнічний комплекс

СУ – система управління

ТП – точка перетину

ТРП – термінальний пристрій

# ВСТУП

На даний час в різних областях людського життя широко застосовуються промислові маніпуляційні роботи (МР). Їх застосування дозволяє підвищити продуктивність технологічних процесів, проводити операції в агресивних середовищах та усунути присутність людей на небезпечних ділянках.

МР це складний електромеханічний об'єкт, що складається з маніпулятора і перепрограмованої системи управління та має низку особливостей:

– МР відрізняються кінематичною структурою, яка містить безліч незалежних або взаємопов'язаних ланок;

– зміна положення взаємопов'язаної ланки в просторі впливає на фізичні сили, що діють на маніпулятор;

– необхідна синхронність управління великим числом двигунів.

У зв'язку з наявністю зазначених особливостей, для інтеграції МР в виробничі процеси потрібні спеціальні системи управління (СУ). Вони необхідні для взаємодії між МР та оператором і забезпечують виконання необхідних процесів автоматизації технологічної операції.

Безліч використовуваних у вітчизняній промисловості СУ МР – це закордонні розробки. До того ж вони є закритими і пропрієтарними рішеннями, тому виникає залежність від іноземних фірм в поставки і технічної підтримки СУ і їх компонентів. Так само обмежуються можливості по їх адаптації при вирішенні специфічних завдань, так як їх використання визначається функціональними рішеннями, закладеними виробником.

У зв'язку з цим виникає проблема імпортозаміщення. Існуючі вітчизняні розробки, на даний час технічно застарілі. Таким чином, завдання розробки сучасної СУ МР, відповідної закордонним аналогам, є актуальною проблемою.

*Об’єкт дослідження* – маніпуляційний робот.

*Предмет дослідження* – підвищення ефективності функціонування інтелектуальної системи управління маніпуляційним роботом.

*Методи дослідження* – дослідження системи виконувалося методами імітаційного моделювання.

Мета роботи – забезпечення якості управління маніпуляційним роботом на основі розробки інтелектуальної системи підтримки прийняття рішень.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати загальні підходи і визначити вимоги що до управління маніпуляційним роботом;

– дослідити особливості кінематики маніпулятора і сформувати його математичну модель;

– розробити і дослідити систему управління маніпуляційним роботом;

– розробити програмні засоби СУ МР;

– змоделювати МР;

– провести дослідження розробленої СУ МР.

Атестаційна робота виконана згідно ДСТУ 3008:2015 [1], керуючись навчальним посібником з дипломного проектування [2] та методичними вказівками [3].

Для вирішення поставлених завдань використані матричне і операційне числення, структурні схеми, методи просторових перетворень і об'єктно-орієнтованого програмування.

В ході виконання магістерської атестаційної роботи підготовлені публікації [4], [5].

# 1 АНАЛІЗ СИСТЕМ МОДЕЛЮВАННЯ МАНІПУЛЯЦИЙНИХ РОБОТІВ

## 1.1 Аналіз конструктивних особливостей промислових маніпуляторів

Маніпуляційний робот складається з механічного маніпулятора і перепрограмованої системи керування, призначений для здійснення певних, наперед заданих переміщень деталей, матеріалів, інструментів або спеціальних пристосувань з метою виконання різних робіт. Важливою частиною МР є маніпулятор – це компонент для виконання рухових функцій, аналогічних функціям руки людину при переміщенні об'єктів в просторі, який має робочий орган [6].

Маніпулятор може застосовуватися в різних областях, тому можуть використовуватися різні схеми побудови механічної частини. Слід розрізняти структури «руки» і «зап'ястя».

Структура «руки» це послідовність ланок, з'єднаних між собою обертальними і поступальними зв'язками. За характером і кількістю зв'язків більшість роботів відносять до однієї з чотирьох категорій [7, 8]:

– з декартовой системою координат (3П) (рис. 1.1, а);

– з циліндричної системою координат (В3П) (рис. 1.1, б).

– зі сферичною системою координат (2ВП) (рис. 1.1, в);

– з обертальної системою координат (nВ) (рис. 1.1, г).

Також кожна структура має свою кількість ступенів вільності. Все залежить від певної задачі маніпулятора. На прикладі робота PUMA, який буде розглянутий надалі, будуть проходити розрахунки. Він має обертальну систему координат та 6 ступенів вільності.

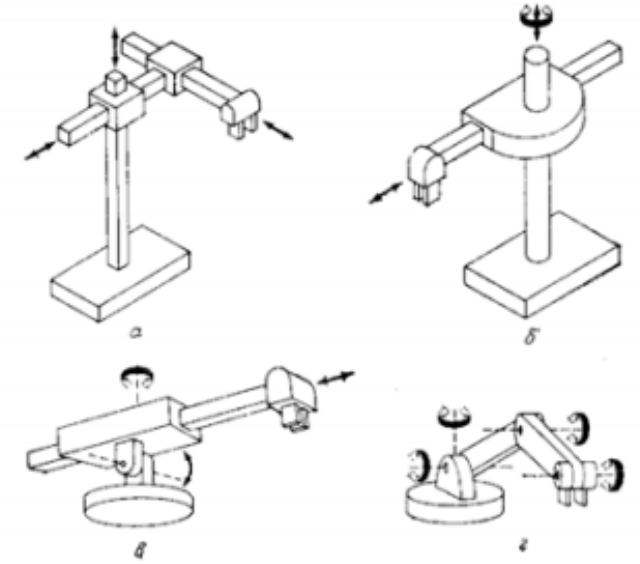


Рисунок 1.1 – Конструкції маніпуляційних роботів

Також сьогодні розвиваються ще 2 типу мапінуляторов: SCARA–маніпулятори (Selective Compliant Assembly Robot Arm) (рис. 1.2, б) та які використовують паралельні зв'язки (рис. 1.2, а) [9].

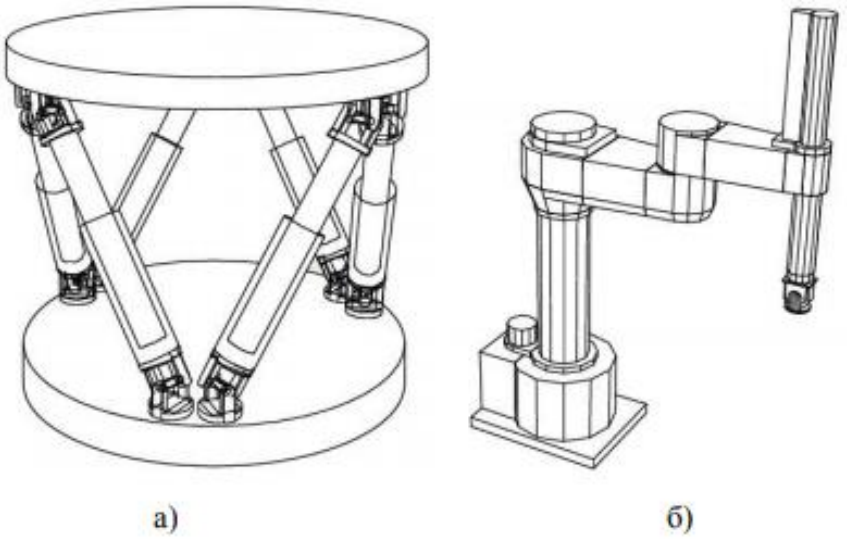



Рисунок 1.2 – Конструкція маніпулятора SCARA-маніпулятора та з паралельними зв'язками

Вибір конкретної реалізації і конструкції МР визначається, перш за все, областю його безпосереднього застосування. Основними напрямками їх використання в даний час є (рис. 1.3):

– промисловість;

– сільське господарство;

– космічна техніка;

– медицина.

Із зазначених областей застосування найбільшого поширення набула промисловість.

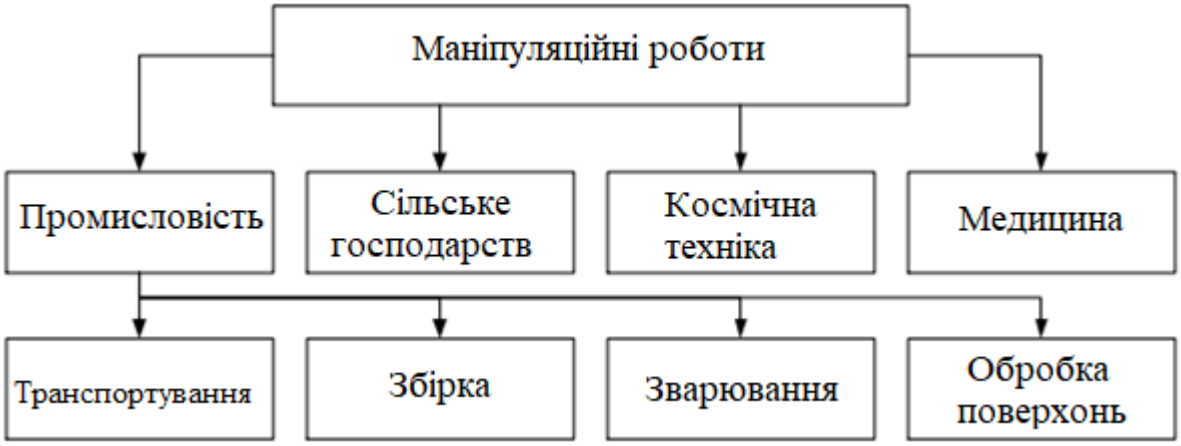



Рисунок 1.3 – Області застосування маніпуляційних роботів

Можна виділити чотири основні класи операцій, виконуваних за допомогою МР [10, 11]:

– транспортування, упаковка, палетування;

– збірка;

– зварювання та різання;

– обробка поверхонь.

У разі, якщо маніпулятор використовується для транспортування деталей, він є частиною більш складної технологічної операції, наприклад, обробки на металообробному верстаті.

Як приклад завдання, в якій раціонально використання МР, можна привести виробничу операцію, пов'язану з обробкою на верстаті (рис. 1.4). Маніпулятор повинен взяти деталь зі спеціально розташованого піддону, помістити її в зону обробки, враховуючи обмеження робочої зони, перемістити виріб по заданій траєкторії руху для обробки, та по завершенню роботи доставити в зону готових виробів.

Рисунок 1.4 – Обробка виробів на верстаті з використанням маніпуляційного робота на скляному заводі в м Гусь-Хрустальний

Конкретним прикладом може служити МР, що здійснює переміщення деталей на конвеєр (рис. 1.5). Дана операція полягає у визначенні положення наступного об'єкта, його захоплення і переміщення на рухому поверхню.

Рисунок 1.5 – Транспортний маніпулятор

Значна маса переміщуваних об'єктів призводить до збільшення впливу динамічних факторів на переміщення маніпулятора, що вимагає ускладнення структури управління МР і підвищення вимог до продуктивності.

Згідно з проведеним аналізом, основними завданнями маніпуляційного робота є позиціонування робочого органу і дотримання заданої траєкторії.

Позиціонування або переміщення маніпулятора в задану точку без необхідності проходження певної траєкторії вимагає врахування насамперед кінематичної структури маніпулятора. В даному випадку динамічні характеристики, під якими розуміються фізичні сили, що діють на маніпулятор в цілому і окремі його компоненти, не роблять істотного впливу на якість виконання технологічної операції. Сучасні загальнопромислові маніпулятори забезпечують точність позиціонування порядку 0,05 – 0,1 мм. З огляду на необхідність скорочення часу, що витрачається на переміщення і позиціонування, необхідно визначити підходи до формування траєкторії і профілю швидкості (рис. 1.6) [12].

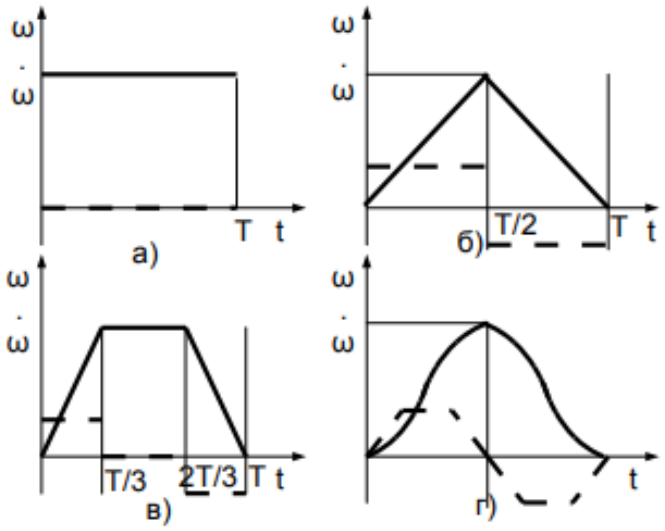



Рисунок 1.6 – Варіанти профілю швидкості переміщення

З наведених графіків бачимо, що в залежності від обраного закону зміни швидкості змінюється співвідношення час / швидкість виконання операції, а також крива прискорення (показана пунктиром). Як видно з рис. 1.6 трикутний (рис. 1.6, б) і трапецеїдальний (рис. 1.6, в) профілі призводять до виникнення зламу кривої прискорення в місці зміни. Використання S-кривої (рис. 1.6, г) для формування швидкості дозволяє сформувати гладку криву переміщення без виникнення різких змін швидкості і прискорення. Таке рішення дозволить знизити навантаження на механічні та електричні компоненти маніпулятора, тим самим, збільшивши термін його експлуатації.

Рух по заданій траєкторії можна розглядати, як сукупність безлічі малих переміщень між окремими точками. Виходячи з цього сказане вище про профіль швидкості вірно і для даного завдання. Слід зазначити, що в даному випадку закони зміни швидкості і прискорення повинні бути визначені виходячи з вимог дотримання заданої точності позиціювання в будь-якій точці траєкторії.

Відмінною особливістю контурних переміщень є істотний вплив динамічних характеристик маніпулятора на якість управління рухом. Перш за

все, це проявляється в зміні фізичних параметрів окремих ланок маніпулятора (моменти інерції щодо зчленувань, сили взаємодій окремих ланок).

При розробці системи управління маніпулятора суттєвою проблемою є необхідність врахування нелінійних елементів в його кінематичной структурі (рис. 1.7) [13]. Основним з них виявляється вплив сухого і в'язкого тертя. Також необхідний облік пружних зв'язків між окремими ланками і в механічних передачах від двигунів до зчленування. Основними видами механічної передачі в МР є зубчаста і ремінна.

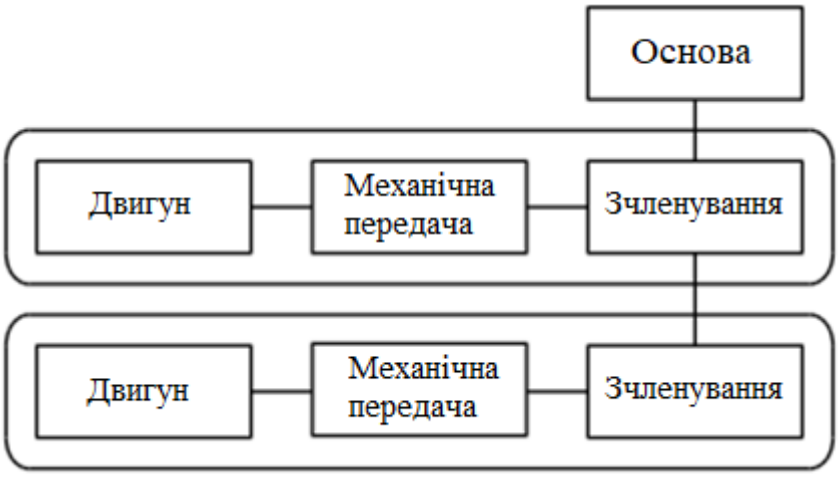



Рисунок 1.7 – Кінематична структура маніпулятора

Якщо розглядати МР в цілому, то вплив кінематичних нелінійностей виявляється значно меншим у порівнянні з динамічними [14].

Для управління МР необхідна наявність вимірювальних датчиків для визначення змінних стану його електромеханічної моделі. Основними вимірюваними змінними для кожної ланки є:

– кутова координата;

– кутова швидкість;

– струм навантаження.

Кутові величини дозволяють визначити положення ланки МР в просторі. Визначення лінійних координат вимагає вирішення завдань кінематики для перекладу кутових координат ланок в набір декартових координат і кутів Ейлера, що визначають поточний стан і орієнтацію робочого органу в просторі. Визначення точності позиціонування для маніпуляційних роботів, як правило, нормується в лінійних координатах [15]. Для більшості технологічних операцій необхідна точність варіюється в межах 0,1 – 1 мм.

Крім положення і швидкості для управління маніпулятором може знадобитися вимір моментів сил, що діють на ланки. Це найбільш важливо для силомоментного управління.

Вимірювання струму навантаження двигуна дозволяє побічно визначити момент на валу двигуна і зробити на основі отриманої інформації обчислення динамічних характеристик маніпулятора. Вимірювання струму і перерахунок його в момент дозволяє уникнути необхідності використання спеціальних датчиків моменту.

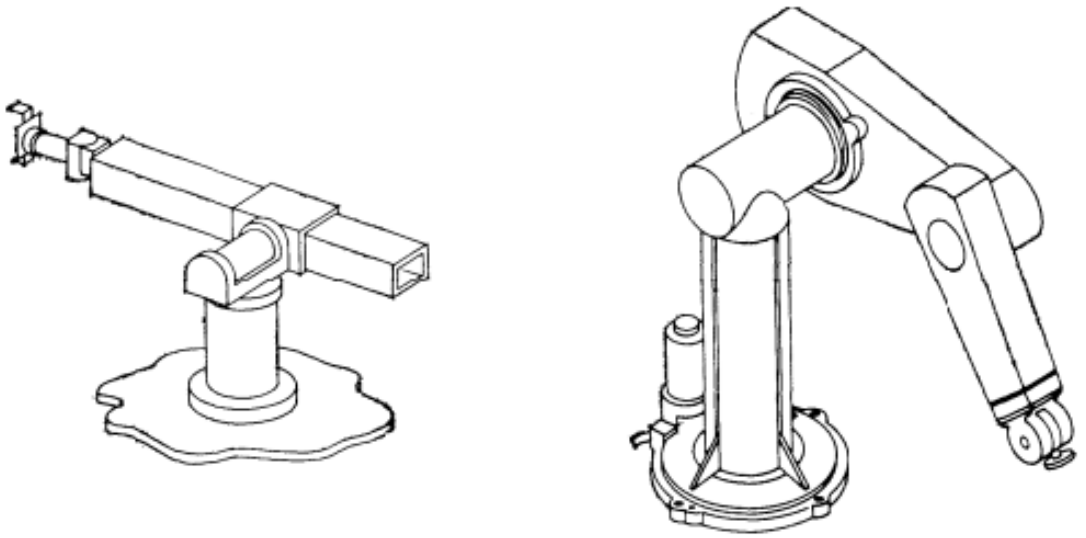
На характеристики систем управління складними мехатроними об'єктами, такими як маніпуляційні роботи, істотний вплив роблять їх кінематичні та динамічні параметри. Внаслідок цього необхідно дати їх докладний математичний опис, зручне для використання при реалізації СУ.

У вирішенні зазначеної задачі можна виділити два взаємопов'язаних напрямки. Перше полягає в створенні точної кінематичної моделі маніпулятора, що дозволяє однозначно визначити його просторову конфігурацію, що в свою чергу дасть можливість описувати закони переміщення робочого органу. Другим напрямком є опис динамічних характеристик і зв'язків, що існують в маніпуляторі, що дозволить описати його поведінку при переміщенні по заданій траєкторії.

## 1.2 Кінематика маніпуляційних роботів

Опис кінематичної схеми маніпулятора вимагає знання його габаритних розмірів і параметрів робочого простору.

Різні конфігурації МР відрізняються за своєю структурою і характеристиками [7]. Їх можна класифікувати за типом використовуваних кінематичних зв'язків, як зазначено в розділі 1.1. У промисловості найбільшого поширення набули багатоланкові МР, що володіють прогресивними і обертальними зчленуваннями (рис. 1.8).



а) б)

Рисунок 1.8 – Багатоланкові маніпуляційні роботи:
а) стенфордський маніпулятор;
б) маніпуляційний робот PUMA.

З проведеного в розділі 1.1 аналізу випливає, що маніпулятори використовують різні координатні системи, кожна з яких має особливості математичного опису. Для того щоб уникнути неоднозначностей при розробці

СУ слід визначити систему координат для опису переміщень. У разі МР, що використовується у виробництві, найбільш раціональним варіантом є використання декартової системи координат (для визначення положення) і кутів Ейлера (для вказівки орієнтації маніпулятора). Таким чином, основним завданням опису кінематичних характеристик манпулятора є перетворення між його власною і обраної рабчей координатної системою.

Для планування траєкторії руху маніпулятора і визначення його положення в просторі необхідно вирішити два основні класи задач у вигляді прямої і зворотної задач кінематики [16].

Рішення прямої задачі служить для перетворення інформації про полжение маніпулятора з власної КС в робочу (абсолютну), що вимагає для визначення координат робочого органу маніпулятора.

Рішення оберненої задачі призначене для обчислення необхідної просторової конфігурації маніпулятора по положенню робочого органу, і є основною проблемою при плануванні траєкторії його переміщення.

Вирішення зазначених завдань вимагає опису габаритних характеристик маніпулятора в формі, зручній для їх аналізу і запису рівнянь перетворення координат. З існуючих підходів основними є вираз їх у вигляді системи лінійних або матричних рівнянь [17].

Для виконання чисельних перетворень найбільш ефективним є метод однорідних перетворень. Стосовно до опису маніпуляційних роботів широкого поширення набуло уявлення Денавіта-Хартенберга (ДХ-уявлення). Воно дозволяє записати кинематику маніпулятора деяким набором матриць просторових перетворень розмірності 4x4, що виражають відносне положення КС окремих ланок.

В цьому випадку кінематична схема маніпулятора описується рівнянням виду:

$$T = \prod_{i=1}^{n} A_i^{i-1}, \quad (1.1)$$

де T – матриця положення робочого органу;

Ai – матриці перетворення і-зчленування;

n – кількість зчленувань в МР.

Рішення прямої задачі кінематики вимагає перетворення безлічі координат, що описують стан маніпулятора в його власній КС, в координати робочої КС.

$$(q_1, q_2, \ldots, q_{1n}) \rightarrow (x_1, x_2, \ldots, x_n),$$

де $q_1, q_2, \ldots, q_{1n}$ – координати у власній системі координат (приєднані);

$x_1, x_2, \ldots, x_n$ – координати в абсолютній системі координат (абсолютні).

При використанні маніпулятора PUMA, який має шість ланок та працює у обертальної системі координат, і застосуванні раніше обраної робочої системи координат, скориставшись виразом (1.1), отримаємо:

$$T = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 A_6^5 = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1.2)$$

де n, s, a – вектори 3x1 орієнтації робочого органу;

p – вектор 3x1 положення робочого органу в базовій системі координат.

На відміну від прямої задачі кінематики зворотна має значно більше підходів до свого рішення, з яких основними є методи зворотних перетворень і бікватерніонів [18].

Метод зворотних перетворень полягає у визначенні значень окремих приєднаних координат, грунтуючись на існуючих матрицях перетворень (1.2). В результаті можуть бути отримані готові аналітичні вирази.

Рішення задач кінематики за допомогою бікватерніонов здійснюється шляхом подання кінематичної конфігурації через подвійні кватерніони і обчислення значень приєднаних координат за допомогою апарату кватерніони перетворень. Даний метод дозволяє в більш компактній формі в порівнянні з матрицями (1.2) описати кінематичну конфігурацію маніпулятора, але вимагає реалізації більш складних обчислювальних алгоритмів у порівнянні з методом зворотних перетворень.

Виходячи з необхідності реалізації алгоритму розв'язання оберненої задачі кінематики в СУ МР, що володіє обмеженими обчислювальними можливостями, необхідно вибрати підхід з можливістю оптимізації обчислень.

Зворотна задача призводить до виникнення в ряді випадків множинності рішень [7]. Це відбувається у зв'язку з тим, що рішення рівнянь зворотної кінематики в тригонометричних функціях може дати кілька рівнозначних рішень. Щоб уникнути цього, в розрахункові вирази вводяться геометричні змінні, що визначають бажану просторову конфігурацію маніпулятора:

$$\text{РУКА}=\text{sign}(-d_4S_{23}-a_{23}C_{23}-a_2C_2)$$

$$\text{ЛІКОТЬ}=\text{РУКА}*\text{sign}(d_4C_3-a_3S_3)$$

$$\text{ЗАП'ЯСТІ}=\begin{cases}\text{sign}(s*z_4), s*z_4\neq 0,\\ \text{sign}(n*z_4),\ n*z_4=0\end{cases}$$

Наведені вирази дозволяють точно визначити необхідне положення ланок маніпулятора виходячи з його бажаних абсолютних координат і просторової конфігурації. Таким чином, вони є рішенням оберненої задачі кінематики та дозволяють планувальником траєкторії сформувати завдання для переміщення окремих ланок.

## 1.3 Динаміка маніпуляторів

При переміщеннях маніпулятора виникають зміни його просторової конфігурації, що в свою чергу призводить до зміни сил, що діють на його окремі елементи. Для обліку цих змін потрібно описати динамічні характеристики МР і їх зв'язок з кінематичною структурою маніпулятора.

Завдання опису динамічних характеристик МР можна розбити на наступні етапи:

– визначення моментів, що діють на окремі ланки;

– опис сукупності зв'язків між окремими ланками;

– уявлення динамічних характеристик в зручній для подальшого аналізу формі.

У процесі руху сили, що діють на маніпуляційний робот, створюють два основні види моментів:

– гравітаційні моменти;

– відцентрові і коріолісові моменти.

Гравітаційні моменти визначаються впливом сили тяжіння на ланки маніпулятора. Вони діють незалежно від швидкості і є вираженням потенційної енергії маніпулятора. Залежать від маси, положення і орієнтації маніпулятора в просторі.

Відцентрові і коріолісові моменти виникають при переміщенні маніпулятора в просторі. Так як між окремими ланками існують кінематичні зв'язки, сили, що діють на них, визначаються не тільки їх власними швидкостями, але і швидкостями інших ланок.

Узагальнені моменти динамічного взаємодії виникають внаслідок зміни положення ланок при їх прискореному русі один щодо одного.

Загальне рівняння динаміки маніпулятора може бути записано у вигляді [19]:

$$\tau(t)=D\big(q(t)\big)\ddot{q}(t)+h\big(q(t),\dot{q}(t)\big)+c\big(q(t)\big),$$

де $\tau(t)$ – вектор $n\times1$ узагальнених моментів у зчленуваннях маніпулятора;

$D$ – матриця $n\times n$ інерції (кінетичної енергії) маніпулятора;

$q(t)$ – вектор $n\times1$ приєднаних змінних маніпулятора;

$\ddot{q}(t)$ – вектор $n\times1$ узагальнених прискорень;

$h$ – вектор $n\times1$ коріолісових і відцентрових моментів;

$\dot{q}(t)$ – вектор $n\times1$ узагальнених швидкостей;

$c$ – вектор $n\times1$ гравітаційних моментів;

$n$ – кількість зчленувань в МР.

Для визначення гравітаційної складової моменту необхідно описати кінематичну структуру маніпулятора і визначити положення центрів ваги окремих ланок.

Елементи вектора $h$ визначають сумарний момент, створюваний коріолісовими і відцентровими силами, які впливають на окрему ланку.

Матриця $D$ описує рівняння кінетичної енергії маніпулятора, що дозволяє врахувати зміну узагальнених швидкостей переміщення, і є вираженням власних моментів інерції окремих ланок маніпулятора.

Складність при створенні динамічної моделі МР полягає в необхідності запису рівнянь динаміки руху маніпулятора в формі, найбільш підходящою для її подальшої реалізації в СУ. Більшість існуючих методів грунтується на рівняннях Лагранжа, Ньютона-Ейлера, Д'Аламбера, Аппеля [20]. Розглянемо основні особливості кожного з них.

Метод Лагранжа дозволяє створити точний аналітичний опис динаміки МР у вигляді системи диференціальних рівнянь Лагранжа-Ейлера, яка може бути представлена у векторно-матричної формі після опису кінематики маніпулятора у вигляді ДХ-уявлення. Безпосередня реалізація рівняння вимагає значних обчислювальних витрат. У зв'язку з великим обсягом обчислень, цей метод

непридатний для реалізації в системі управління, що працює в режимі реального часу. Тому в таких завданнях необхідно використовувати модифіковані методи [21].

Метод Ньютона-Ейлера полягає в заміні рівнянь Лагранжа-Ейлера системою рекурентних рівнянь, що дозволяє значно скоротити необхідні обчислення. Недоліком цього методу є те, що з його допомогою неможливо врахувати вплив внутрішніх і зовнішніх чинників.

Метод, заснований на використанні принципу Д'Аламбера, полягає в заміні динамічних сил, що діють на ланки маніпулятора додатковими силами інерції, що виражають їх прискорений рух відносно один одного і в просторі. Це робить його зручним для аналізу впливу окремих сил на маніпулятор. При цьому для реалізації даного методу потрібно менше обчислювальних витрат в порівнянні з використанням рівнянь Лагранжа-Ейлера. Однак, сили інерції розглядаються як єдине ціле, що не дозволяє оцінити вплив їх окремих компонентів.

Використання рівнянь Аппеля дозволяє записати рівняння динаміки у вигляді диференціальних рівнянь в системі псевдокоординатах. Даний метод передбачає, що сили, що діють на ланки маніпулятора, є ідеальними. Це дозволяє зробити припущення про вплив внутрішніх сил взаємодії, але не дає можливість врахувати можливі зовнішні чинники.

### 1.4 Висновки

Маніпуляційні роботи все більше застосовуються в різних сферах людської життєдіяльності. Вони дозволяють виконувати роботу швидше, точніше, якісніше. Але для будь-якого МР потрібна система управління. До завдань СУ входить: пошук маршруту, рішення кінематичних завдань і облік динаміки маніпулятора. Інтелектуальна система прийняття рішень дозволяє ефективно вирішувати завдання з пошуку маршруту .

# 2 РОЗРОБКА СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ ЩОДО АДАПТАЦІЇ ПОЛОЖЕНЬ МАНІПУЛЯЦІЙНОГО РОБОТА

## 2.1 Опис маніпулятора та робочого простору

Грунтуючись на проведеному аналізі, можна визначити загальні структурні рішення (рис. 2.1), використовувані в сучасних СУ МР, і виділити наступні компоненти:

– засоби програмного управління, призначені для управління з боку кінцевого користувача і рішення технологічних завдань;

– засоби адаптивного управління, необхідні для управління маніпулятором в умовах змінюючихся зовнішніх впливів і параметрів

– приводи, які здійснюють безпосереднє управління ланками маніпулятора

– засоби очувствленія для вирішення складних технологічних завдань, що вимагають розгорнутої інформації про стан робочого простору.

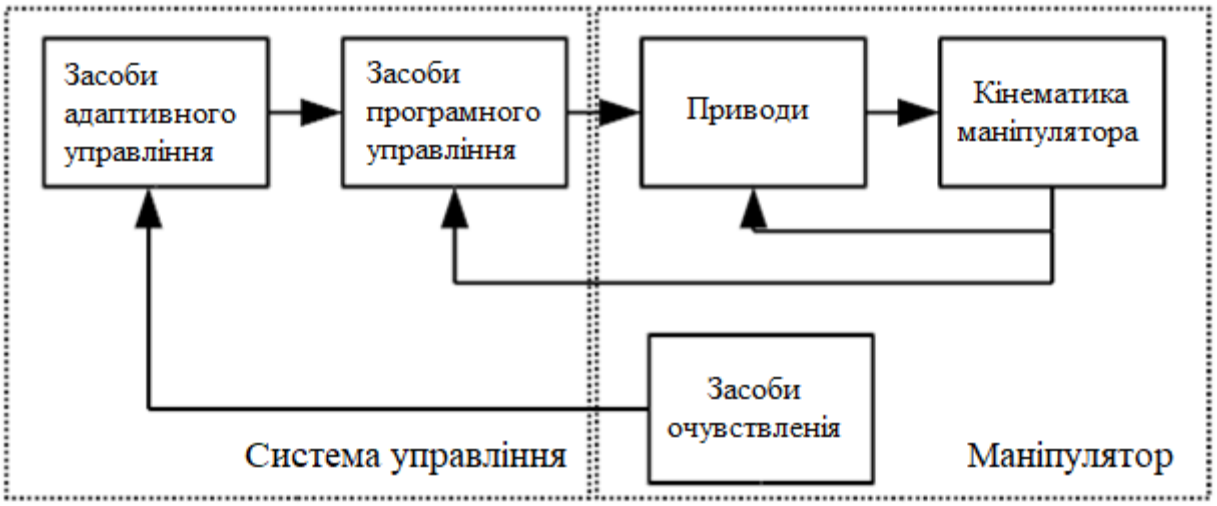



Рисунок 2.1 – Структурна схема СУ МР

Засоби програмного управління включають базовий набір алгоритмів, призначених для управління електромеханічними компонентами маніпулятора, в першу чергу двигунами осей, а також забезпечують можливість завдання керуючої програми кінцевого користувача.

Також сюди відносяться різні алгоритми інтерполяції траєкторії, в тому числі лінійна, кругова і сплайнова.

Засоби адаптивного управління включають засоби, необхідні для обліку змінюючихся параметрів маніпулятора і навколишнього середовища. Мінімальною вимогою в даному випадку є облік зміни динамічних характеристик в результаті його переміщення [14].

Таким чином, обчислення, що виконуються СУ, є досить складними, оскільки вимагають проведення матричних операцій високого порядку, рішення задач кінематики та динаміки, а також обчислення змінних стану МР на підставі даних вимірювань для розрахунку регулятора [22].

Для забезпечення необхідної якості управління СУ МР повинна враховувати різні внутрішні і зовнішні чинники (зміна просторової конфігурації маніпулятора, маси і конфігурації навісного обладнання або деталі). Для цього може знадобитися введення алгоритмів управління, які дозволяють компенсувати їх вплив на маніпулятор. В якості базової функціональності в СУ МР повинна бути передбачена можливість розрахунку кінематичної і динамічної моделей маніпулятора з метою мінімізації впливу внутрішніх факторів, таких як взаємовплив ланок і зміна їх моментів інерції при переміщенні.

Крім зазначених вимог, існує ряд факторів, облік яких дозволяє підвищити зручність застосування СУ МР і розширити можливості її експлуатації.

Для виконання керуючої програми СУ МР повинна надавати функціональні можливості по створенню, зберіганню і виконанню керуючої програми. Для цього необхідні засоби для введення інформації в СУ, обміну з

користувачем і управління ходом відпрацювання УП. Відсутність єдиного стандарту написання УП вимагає розробки власних підходів до програмування.

Інтерфейс повинен дозволяти оператору спостерігати за роботою СУ МР, виконувати необхідні операції для підтримки її в робочому стані (запуск / зупинка програми, спостереження за режимами і параметрами робототехнологічних комплексу (РТК). Робота РТК може здійснюватися в трьох основних режимах [23]:

– ручному – маніпулятор управляється користувачем;

– автоматичному – виконується програма;

– навчання – проводиться запис керуючої програми.

Для того, щоб СУ відповідала вимогам, що пред'являються сучасним виробництвом, вона повинна бути адаптованою до застосування в технологічних операціях, які використовують різні види маніпуляторів і периферійного обладнання. Щоб забезпечити це, в основу розробки СУ МР слід покласти принципи модульності та відкритої архітектури.

Принцип модульності полягає в можливості комплектування кінцевої СУ набором компонентів, які найбільш підходять до виконання поставленого завдання. Основні складові частини, представлені на рис. 2.2, є модулями, склад і реалізація яких можуть визначатися конкретним завданням, виконуваної МР.

Принцип відкритої архітектури полягає в можливості конфігурації і створення окремих компонентів СУ кінцевим користувачем, використанні відкритих протоколів обміну даними і надання доступу до програмних модулів для розробки призначених для користувача програм і алгоритмів.

У структурі програмного забезпечення (рис. 2.2) необхідно виділити два класи: програмний засіб (ПЗ) контролера руху (КР) та термінального пристрою (ТРП).


Рисунок 2.2 – Структура ПЗ КД і ТРП

ПЗ КД включає в себе елементи, призначені для забезпечення роботи БУ і взаємодії з апаратними засобами СУ МР. В роботі приділяється більше уваги плануванню і розрахуноку траєкторії руху.

У набір засобів планування і розрахунку траєкторії руху входять такі компоненти, як планувальник траєкторії (ПТ), програми розрахунку прямої і зворотної задач кінематики, система обробки керуючих програм.

На характеристики систем управління складними мехатронними об'єктами, такими як маніпуляційні роботи, істотний вплив роблять їх кінематичні та динамічні параметри. Внаслідок цього необхідно дати їх докладний математичний опис, зручне для використання при реалізації СУ.

У вирішенні зазначеної задачі можна виділити два взаємопов'язаних напрямки. Перше полягає в створення точної кінематичної моделі маніпулятора, що дозволяє однозначно визначити його просторову конфігурацію, що в свою чергу дасть можливість описувати закони переміщення робочого органу. Другим напрямком є опис динамічних характеристик і зв'язків, що існують в маніпуляторі, що дозволить описати його поведінку при переміщенні по заданій траєкторії.

Опис кінематичної схеми маніпулятора вимагає знання його габаритних розмірів і параметрів робочого простору.

Різні конфігурації МР відрізняються за своєю структурою і характеристиками [7]. Їх можна класифікувати за типом використовуваних кінематичних зв'язків, як зазначено в розділі 1. У промисловості найбільшого поширення набули багатоланкові МР, що володіють прогресивними і обертальними зчленуваннями. Як приклад можна відзначити маніпулятори таких фірм як KUKA, Fanuc, Kawasaki, ABB. Вони найбільш широко поширені на виробництві.

## 2.2 Пошук траєкторій переміщення

Завданням планувальника траєкторії є знаходження найкращого або оптимального маршруту між двома точками простору, який використовуються маніпуляційним роботом. Критерій оптимальності маршруту залежіть від контексту задачі. Це може бути найближчий або найменш енерговитратний шлях. Але найголовнішою задачею є прохідність маршруту.

Пошук маршруту це важлива частина при автоматизації МР. Інтелектуальна система прийняття рішень повинна автоматично знаходити найефективніший шлях. Якщо на обраному маршруті з’являється будь-яка

перешкода – планувальник траєкторії повинен автоматично скоригувати знайдений маршрут [24].

Для пошуку шляху можна виділити три найвідоміших методу:

– A* (A star);

– навігаційних сіток (navigation mesh);

– навігаційний граф.

В алгоритмі A* (A star) розбивається простір на однакові клітини. Для кожної клітини задається властивість: прохідна клітина чи ні. Також займає одну клітку об'єкт, для якого відбувається пошук шляху, в нашому випадку це схват. На основі матриці прохідності знаходиться маршрут з однієї клітини в іншу хвильовим алгоритмом або його модифікацією. На рисунку 2.3 показаний приклад алгоритму A *, за результатами попередніх досліджень [25]. Помаранчеві клітини – це початкова і кінцева точка маршруту, жовті клітини – маршрут, а чорні – перешкоди.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 7 | 8 |
| 6 | 6 | | 5 | 5 | 5 | 5 | 5 | 5 | 6 | 7 | 8 |
| 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 5 | 6 | 7 | 8 |
| 4 | 4 | 5 | 4 | 3 | 3 | 3 | 4 | | | 7 | 8 |
| 3 | | | | | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 1 | 1 | 2 | 3 | 4 | 5 | | | 8 |
| 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | | | 8 |
| 3 | 2 | 1 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Рисунок 2.3 – Приклад алгоритму A* (A star)

Даний метод найбільше використовується в 2-х вимірному просторі (як в бакалаврської роботі, при вирішенні транспортної задачі). Але якщо його розглядати на 3-х мірної площині, то вийде ряд недоліків:

– в 3-х мірному просторі об'єкти часто довільної форми, які неефективно розміщувати в клітинній структурі;

– динамічні об'єкти під час пересувань або поворотів можуть займати надмірна кількість додаткових (сусідніх) клітин;

– переміщення по клітинах в 3-х мірному просторі виглядає неприродно;

– для більш точного позиціонування необхідно збільшувати масштаб, що призводить до значного збільшення тривалості пошуку маршруту.

У методі навігаційних сіток прохідний простір задається у формі полігонів 3-х мірної моделі [26]. Полігон опуклий, а значить між 2 будь-якими вершинами є шлях, який завжди знаходиться всередині обраного полігону. В даному методі важко враховувати динамічні об'єкти, так як обчислювати їх занадто витратно, а отже, потрібні значні ресурси. Залежно від складності оточення і його деталізації може знадобитися велика кількість полігонів, можливо складних форм. Для автоматизації процесів нам може знадобитися автоматична генерація сіток. Операція значно складніша і для деяких випадків результат не зовсім коректний.

Метод навігаційних сіток дозволяє вибрати конкретний підхід в якості контрольних точок шляху:

– через центри полігонів;

– через вершини;

– через центр ребер;

– змішаний обхід.

У прикладі (рис. 2.4) жовта лінія – це знайдений шлях з використанням вершин і центру ребер як контрольних точок маршруту, а зелена лінія – ідеальний маршрут. Знайдений маршрут за прикладом неоптимальний. Для вирішення цієї

проблеми можна збільшувати кількість точок на ребрах, але в такому випадку зросте складність вирішення.

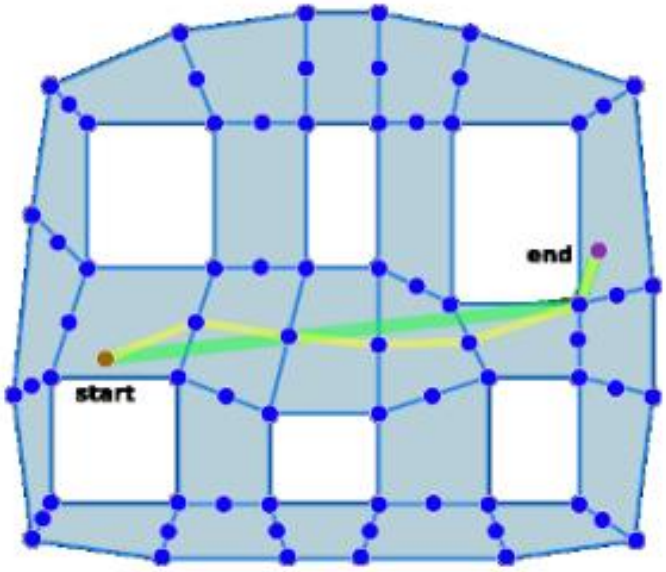



Рисунок 2.4 – Приклад пошуку шляху методом navigation mesh

Найбільш відповідний для нашого випадку це модифікація методу навігаційного графа. Але на початку розглянемо базовий метод і його недоліки.

Метод навігаційного графа (НГ) веде пошук маршруту з використанням графа (рис. 2.5). Відповідно в графі є вершини – точки в 3-х мірному просторі і ребра, які з'єднують ці точки. Вага у ребер відповідає відстані між точками. Кожне ребро має бути проходимо.

На початку йде пошук найближчі вершини графа для стартовою і кінцевої точки. А далі вирішується звичайна задача пошуку шляху на графі між знайденими вершинами, де вага ребер повинна бути самою мінімальною.

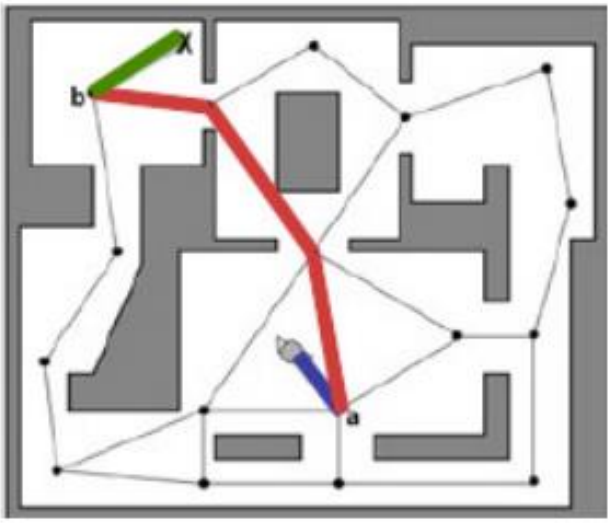



Рисунок 2.5 – Приклад пошуку шляху за допомогою навігаційного графа

Метод НГ хоч і підходить для 3-х мірного простору, але можна виділити ряд недоліків при його використанні:

– для складного об'єкта необхідно більш детальний опис, що збільшує кількість вершин іноді у багато разів;

– в деяких ситуаціях переміщення відбуватиметься по неприродної траєкторії. Це через те, що рух слідує по заданих ребрах графа. Щоб зробити більш природне пересування – необхідно збільшити кількість вершин, але це посилить першу проблему;

– у разі, якщо на ребро навігаційного графа потрапить динамічний об'єкт – ребро стане непрохідним;

– навігаційний граф будується для конкретного об'єкта з певними розмірами, а значить побудований граф не підійде до інших об'єктів різної форми або розміру.

Для нашого випадку з маніпулятором останній недолік опускається, так як переміщення відбуватиметься лише для одного об'єкта – схват маніпулятора. У розділі 2.3 розглянемо детально модифікований метод НГ, який дозволяє вирішити інші недоліки.

Використання програм розрахунку кінематики необхідно для перекладу заданих рухів з координатної системи, яка застосовується в керуючій програмі в координатну систему, використовувану в маніпуляторі, і назад. У зв'язку з тим, що для багатоланкових маніпуляторів потрібне проведення ряду складних просторових перетворень, розрахункові алгоритми повинні бути оптимізовані за швидкодією.

Виходячи з формули (1.2) окремі матриці перетворення приймуть вид:

$$A_1^0=\begin{bmatrix}\cos\theta_1 & 0 & -\sin\theta_1 & 0\\ \sin\theta_1 & 0 & \cos\theta_1 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$A_2^1=\begin{bmatrix}\cos\theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2\\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2\\ 0 & 0 & 1 & d_2\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$A_3^2=\begin{bmatrix}\cos\theta_3 & 0 & \sin\theta_3 & -a_3\cos\theta_3\\ \sin\theta_3 & 0 & -\cos\theta_3 & a_3\sin\theta_3\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$A_4^3=\begin{bmatrix}\cos\theta_4 & 0 & -\sin\theta_4 & 0\\ \sin\theta_4 & 0 & \cos\theta_4 & 0\\ 0 & -1 & 0 & d_4\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$A_5^4=\begin{bmatrix}\cos\theta_5 & 0 & \sin\theta_5 & 0\\ \sin\theta_5 & 0 & -\cos\theta_5 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$A_6^5=\begin{bmatrix}\cos\theta_6 & -\sin\theta_6 & 0 & 0\\ \sin\theta_6 & \cos\theta_6 & 0 & 0\\ 0 & 0 & 1 & d_6\\ 0 & 0 & 0 & 1\end{bmatrix},$$

де $\theta_i$, $d_i$, $a_i$ и $\alpha_i$ – параметри ДХ-уявлення (рис. 2.3);

$\theta_i$ – приєднаний кут, на який потрібно повернути вісь $x_{i-1}$ навколо осі $z_{i-1}$, щоб вона стала сонаправлена з віссю $x_i$;

$d_i$ – відстань між перетином осі $z_{i-1}$ з віссю $x_i$ і початком (i-1)-й системи координат;

$a_i$ – лінійне зміщення (найкоротша відстань між осями $z_{i-1}$ та $z_i$);

$\alpha_i$ – кутовий зсув – кут, на який потрібно повернути вісь $z_{i-1}$ навколо осі $x_i$, щоб вона стала сонаправлена з віссю $z_i$.

Значення параметрів $d_i$, $a_i$ і $\alpha_i$ визначаються орієнтацією абсолютної системи координат і вибором точки відліку (рис. 2.6). Їх визначення і усереднені чисельні вираження наведені в таблиці 2.1 [27].

Величини $\theta_i$ в даному випадку відповідають приєднаним координатами маніпулятора і описують стан маніпулятора у власній системі координат.

Таблиця 2.1 – Параметри ДХ-уявлення маніпулятора PUMA

| | $\theta_i$ | $\alpha_i$ | $a_i$, мм | $d_i$, мм | Межі вимірювання |
|---|---|---|---|---|---|
| I | 90 | -90 | 0 | 0 | -160 +160 |
| II | 0 | 0 | 431,8 | 149,09 | -225 +45 |
| III | 90 | 90 | -20,32 | 0 | -45 +225 |
| IV | 0 | -90 | 0 | 433,07 | -110 +170 |
| V | 0 | 90 | 0 | 0 | -100 +100 |
| VI | 0 | 0 | 0 | 56,25 | -266 + 266 |

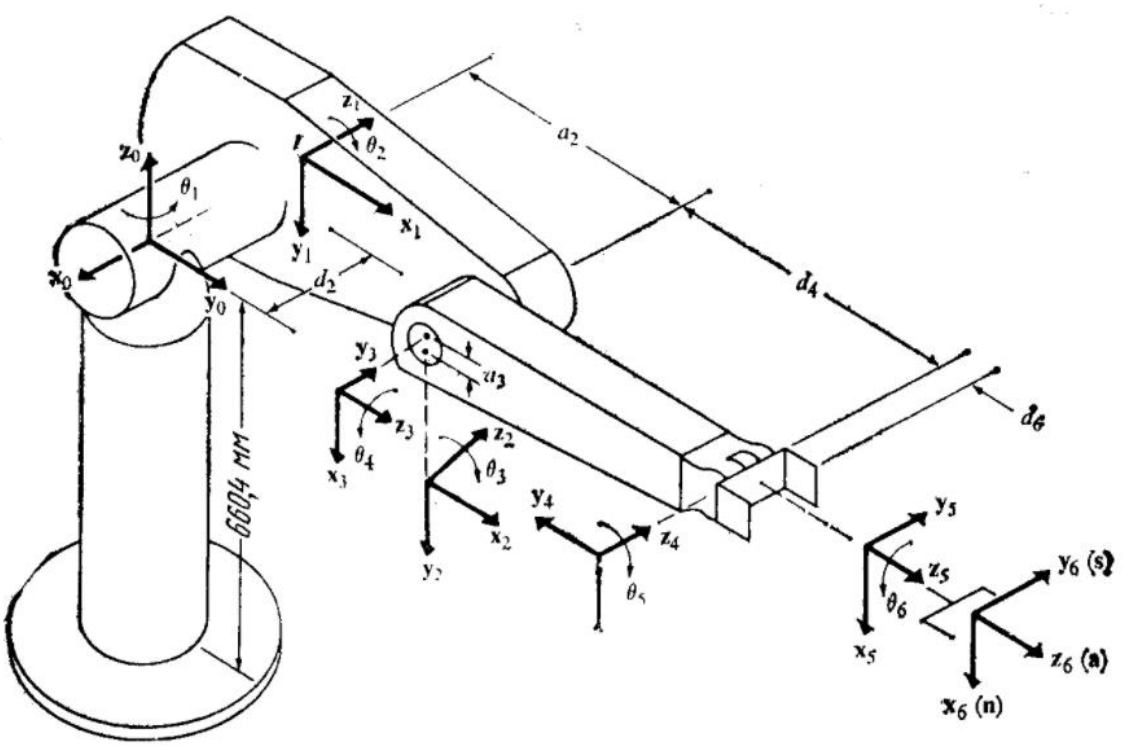



Рисунок 2.6 – Формування систем координат ланок

Для розрахунку координат маніпулятора в умовах, коли матричні операції неприпустимі або нераціональні з точки зору використання обчислювальних ресурсів, можливо опис прямої задачі кінематики в вигляді системи лінійних рівнянь:

$$n_x=C_1[C_{23}(C_4C_5C_6-S_4S_6)-S_{23}S_5C_6]-S_1(S_4C_5C_6+C_4S_6),$$

$$n_y=S_1[S_{23}(C_4C_5C_6-S_4S_6)+C_{23}S_5C_6]-C_1(S_4C_5C_6+C_4S_6),$$

$$n_z=-S_{23}(C_4C_5C_6-S_4S_6)-C_{23}S_5C_6,$$

$$s_x=C_1[-C_{23}(C_4C_5S_6+S_4C_6)+S_{23}S_5S_6]-S_1(-S_4C_5S_6+C_4C_6),$$

$$s_y=S_1[-C_{23}(C_4C_5S_6+S_4C_6)+S_{23}S_5S_6]+C_1(-S_4C_5S_6+C_4S_6),$$

$$s_z=S_{23}(C_4C_5C_6+S_4C_6)+C_{23}S_5C_6,$$

$$a_x=C_1(C_{23}C_4S_5+S_{23}C_5)-S_1S_4S_5,$$

$$a_y=S_1(C_{23}C_4S_5+S_{23}C_5)+C_1S_4S_5,$$

$$a_z=-S_{23}C_4S_5+C_{23}S_5,$$

$$p_x=C_1[d_6(C_{23}C_4C_5+S_{23}C_5)+S_{23}d_4+a_3C_{23}+a_2C_2]-S_1(d_6S_4S_5+d_2),$$

$$p_y=S_1[d_6(C_{23}C_4C_5+S_{23}C_5)+S_{23}d_4+a_3C_{23}+a_2C_2]+C_1(d_6S_4S_5+d_2),$$

$$p_z=d_6(C_{23}C_5-S_{23}C_4S_5)+C_{23}d_4-a_3S_{23}-a_2S_2,$$

де

$$C_i=\cos\theta_i,\ S_i=\sin\theta_i,$$

$$C_{ij}=\cos(\theta_i+\theta_j),\quad S_{ij}=\sin(\theta_i+\theta_j).$$

Такий підхід може бути використаний, зокрема, при реалізації СУ МР на мікроконтролері, що не володіє достатніми обчислювальними можливостями для виконання складних математичних перетворень.

Повертаючись до розв'язання оберненої задачі кінематики, виходячи з обраного методу та описаних геометричних змінних в розділі 1.3, остаточні вирази для визначення приєднаних координат перших трьох зчленувань МР PUMA приймуть вигляд [7]:

$$\theta_1=\operatorname{arctg}\frac{-\text{РУКА}p_y\sqrt{p_x^2+p_y^2-d_2^2}-p_xd_2}{-\text{РУКА}p_x\sqrt{p_x^2+p_y^2-d_2^2}-p_yd_2},$$

$$R=\sqrt{p_x^2+p_y^2+p_z^2-d_2^2},\ r=\sqrt{p_x^2+p_y^2-d_2^2},$$

$$\sin\alpha=-\frac{z}{R},\ \cos\alpha=-\text{РУКА}\frac{r}{R},$$

$$\cos\beta=\frac{a_2^2+R_2^2-(d_4^2+a_3^2)}{2a_2R},\ \sin\beta=\sqrt{1-\cos^2\beta},$$

$$\theta_2=\operatorname{arctg}\frac{\sin\alpha\cos\beta+\text{РУКА}\times\text{ЛІКОТЬ}\times\cos\alpha\sin\beta}{\cos\alpha\cos\beta+\text{РУКА}\times\text{ЛІКОТЬ}\times\sin\alpha\sin\beta},$$

$$\cos\varphi=\frac{a_2^2+(d_4^2+a_3^2)-R^2}{2a_2\sqrt{d_4^2+a_3^2}},\ \sin\varphi=\text{РУКА}\times\text{ЛІКОТЬ}\sqrt{1-\cos^2\varphi},$$

$$\sin\beta=\frac{d_4}{\sqrt{d_4^2+a_3^2}},\ \cos\beta=\frac{|a_3|}{\sqrt{d_4^2+a_3^2}},$$

$$\theta_3=\operatorname{arctg}\frac{\sin\varphi\cos\beta-\cos\varphi\sin\beta}{\cos\varphi\cos\beta+\sin\varphi\sin\beta},$$

Наведені вирази дозволяють точно визначити необхідне положення ланок маніпулятора виходячи з його бажаних абсолютних координат і просторової конфігурації. Таким чином, вони є рішенням оберненої задачі кінематики та дозволяють планувальником траєкторії сформувати завдання для переміщення окремих ланок.

### 2.3 Застосування модифікованого методу навігаційних графів

Використання модифікованого методу навігаційного графа дозволило розробити систему пошуку ефективного маршруту в 3-х мірному просторі. Для пошуку використовується безліч графів:

$$A=\{A_i, i=1..n\},$$

де $A_i$ – навігаційний граф конкретного об'єкта, заданий множиною вершин V та безліччю ребер R:

$$A_i=\{V, R\},$$
$$V=\{V_k, k=1..s\},$$
$$R=\{R_j, j=1..m\},$$
$$R_j=\{V_{j1}\in V, V_{j2}\in V, T_a, T_d, j1\neq j2\}$$

де $T_a \in \{0, 1\}$ – ознака прохідності,

$T_d$ – контекстна інформація про прохідності простору навколо ребра.

Пошук оптимального маршруту починається з визначення початкової та кінцевої точки $P_н$ і $P_к$. Поєднавши ці точки, вийде відрізок (найкоротший шлях), з яким і будуть проходити розрахунки. Нижче представлений весь алгоритм модифікованого методу НГ:

1. Пошук динамічних об'єктів, які змінили свій стан: позицію чи кут. Якщо була зміна і при цьому об'єкт знаходився в складі об'єднаного графа, то розформовуємо об'єднаний граф на його складові.

$$A_{изм}=\{A_i,\ p_i \neq p_i^{`} \cup a_i \neq a_i^{`}\},$$

де $p_i$ и $q_i$ – позиція і поворот об'єкта в поточний момент часу,

$a_i^{`}$ и $a_i^{`}$ – позиція і поворот в момент попереднього пошуку шляху.

Знайдені об'єкти $A_{изм}$ змінили свій стан, а отже, потрібно перерахувати їх графи;

2. Пошук і об'єднання динамічних об'єктів $A_{пер}$, у яких навігаційні графи перетинаються один з одним. Надалі використовуються об'єднані навігаційні графи, замість пересічних;

3. Пошук навігаційних графів, які перетинаються з відрізком шляху ($P_н$ – $P_к$). Далі знаходяться точки перетину з ребрами графа;

4. Безліч точок перетину упорядковується відповідно до віддаленості від початкової точки маршруту:

$$P_{тп}=\{P_i, D(P_i,P_н)<D(P_{i+1},P_н), i \in 1..n-1\}$$

де $D(P_i,P_н)$ – відстань від точки $P_i$ до точки $P_н$,

n – кількість точок перетину;

5. Список точок перетину розбивається на кілька частин. Причому в кожній частині йде свій набір точок для певного НГ;

6. Перебираємо новий список частин і в кожній з них окремо ведемо пошук маршруту. Самі частини з'єднуємо між собою прямими відрізками. Під час пошуку маршруту в НГ може бути 2 випадки: в першому випадку тільки 1 точка перетину (ТП), а значить шлях тільки доторкується до НГ і тут не потрібно додаткового пошуку маршруту, а в другому випадку 2 ТП і тут виконується пошук маршруту між двома точками перетину НГ за критерієм мінімальної сумарної довжини ребер. При цьому враховуєтья властивість $T_a$ – прохідності ребра;

7. Об'єднання отриманих маршрутів в один кінцевий маршрут. Беремо знайдені маршрути графа і по черзі з'єднуємо кожен один з одним, поєднуючи крайні точки прямою лінією.

Модифікований алгоритм НГ не поступається в ефективності пошуку оптимального маршруту базового алгоритму НГ, а іноді і виграє. Наприклад, в модифікованому алгоритмі (рис. 2.7, зліва) маршрут виглядає більш природним, на відміну від базового (рис. 2.8, праворуч). Таким чином підвищується адекватність знайдених оптимальних маршрутів у відкритому просторі.

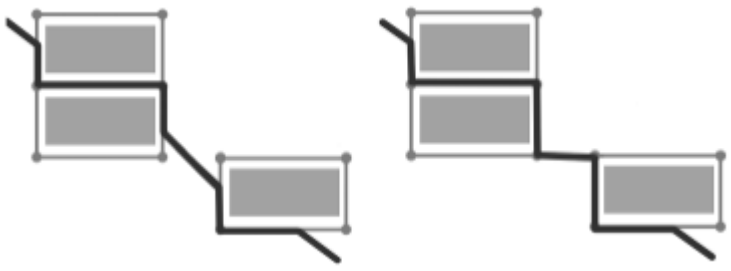


Рисунок 2.7 – Пошук маршруту методом НГ

Таким чином, модифікований алгоритм для пошуку оптимального маршруту на основі методу НГ має свої переваги в ефективності, та позбавляє від недоліків в базовому алгоритмі.

Різниця між статичними і динамічними об'єктами полягає в тому, що динамічні об'єкти можуть змінювати свій стан: позицію і поворот, а також заходити в наш простір або виходити з нього. На момент пошуку маршруту динамічні об'єкти, по суті, не відрізняються від статичних. Але є одна відмінність в розрахунку НГ динамічних об'єктів (перший етап алгоритму).

Через те, що динамічні об'єкти можуть змінювати свій стан: позицію і / або поворот, то з'являється необхідність в механізмі, який буде вирішувати колізії в момент зіткнення з динамічними об'єктами. Одне з рішень це перезапустити заново алгоритм і врахувати новий стан всіх динамічних об'єктів. У той же час нові стани динамічних об'єктів можуть вплинути на НГ статичних об'єктів і деякі ребра можуть стати непрохідні. В цьому випадку вони відзначаються за допомогою властивості $T_a$. Отже ці ребра тимчасово не беруть участь при пошуку маршруту, поки динамічний об'єкт не зрушиться.

Ще один наслідок, що динамічні об'єкти можуть в будь-який час змінювати свій стан – це можливі випадки, коли навігаційні графи об'єктів перетинаються. Причому перетин одного з ребер динамічного об'єкта іншим динамічним об'єктом може спричинити неправильне рішення пошуку маршруту. В цьому випадку необхідно об'єднати 2 графа цих об'єктів, щоб все ребра були прохідними (рис. 2.8).

Нижче представлений алгоритм для об'єднання НГ:

– знаходимо ТП навігаційних графів;

– точки перетину ділять ребра на частини і вже ці частини будуть в ролі нових ребер НГ;

– пошук ребер, які знаходяться всередині НГ. Згодом ці ребра прибираються з масиву ребер НГ.

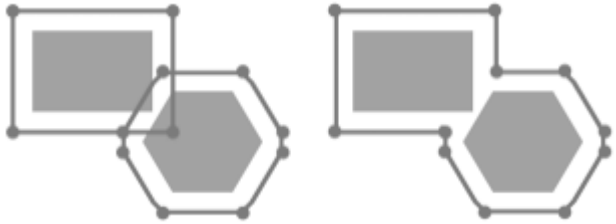


Рисунок 2.8 – НГ динамічних об'єктів до злиття (зліва) і після злиття (праворуч)

Модифікований метод НГ оптимізує генерацію НГ, що значно скорочує обсяг ручної роботи, якщо задається НГ якомусь об'єкту. Таким чином це вирішує одну з проблем в базовому методі НГ.

– позбавляємося від зайвих ребер, допускаючи, що порожній простір без графів завжди проходим в будь-якому напрямку. Тому НГ описуємо для кожного об'єкта окремо (рис. 2.9);

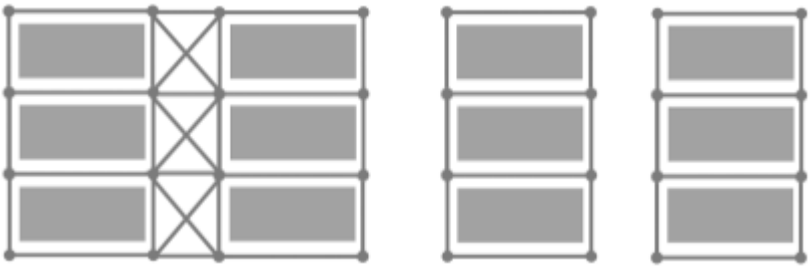


Рисунок 2.9 – НГ в базовому (зліва) і модифікованому (праворуч) методах

– якщо об'єкт повторюється, то НГ для цього об'єкта можна скопіювати, пересунувши точки навігаційного графа за певним законом (рис. 2.10).

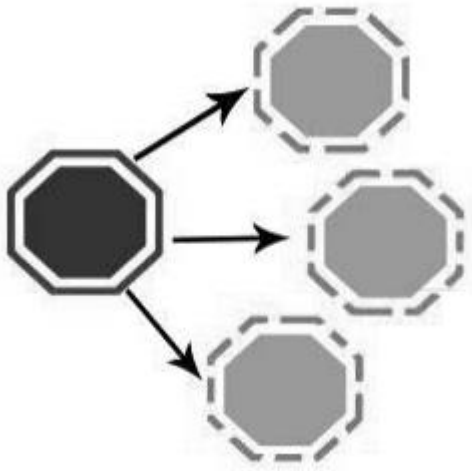


Рисунок 2.10 – Копіювання НГ для повторюваних об'єктів

– генерація НГ для обходу перешкод може бути спрощена, якщо використовується проста геометрична форма перешкоди. Це робиться на основі bounding box об'єкта і даних про його стан.

За рахунок перерахованих особливостей методу значно зменшується обсяг ручної роботи.

Алгоритмічна складність окремих етапів модифікованого методу НГ наведена в таблиці 2.2. Відповідно наведені тільки найвимогливіші етапи. Також в таблиці 2.3 наведена алгоритмічна складність розрахунку динамічних об'єктів.

Таблиця 2.2 – Оцінки складності методу НГ

| № етапу | Алгоритмічна складність |
|---|---|
| 3 | O(N), де N – загальна кількість об'єктів |
| 4 | $O(K_i \log K_i)$, де $K_i$ – кількість точок перетину в і графі |
| 7 | $O(N_i^2)$ |

Таблиця 2.3 – Оцінка складності динамічних об'єктів

| Опис | Алгоритмічна складність |
|---|---|
| Пошук динамічних об'єктів, що змінили стан | O(N), де N – загальна кількість динамічних об'єктів |
| Перерахунок НГ для знайдених об'єктів | $O\left(\sum_{i=1}^{k} \text{len}(V_i)\right)$, де k – число знайдених динамічних об'єктів, len – кількість елементів множини |
| Пошук пересічних об'єктів | O(k*N) |

Модифікований метод НГ вирішує недоліки і проблеми базового методу. В результати виходять більш адекватні маршрути зі зменшеною алгоритмічної складністю.

## 2.4 Висновки

Отже для моделювання інтелектуальної системи прийняття рішень необхідно визначитися з методом пошуку маршруту та вирішити обернену задачу кінематики (ОЗК). За аналізом існуючих методи було обрано модифікований метод НГ. Для рішення ОЗК можна застосувати наприклад метод Девіда-Хантерберга.

# 3 РОЗРОБКА АЛГОРИТМІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ

## 3.1 Розробка алгоритмів

### 3.1.1 Загальний алгоритм

Під час розробки алгоритмів необхідно вирішити два завдання для управління маніпуляційним роботом:

– пошук оптимального маршруту;

– рішення ОЗК.

На початку завантажується карта всіх об'єктів і 2 точки, між якими необхідно знайти оптимальний маршрут. Далі відбувається угруповання всіх об'єктів. Після чого з'єднуються прямою 2 точки, між якими відбувається пошук оптимального шляху і ТП з об'єктами. Якщо були знайдені ТП з об’єктами – відбувається пошук додаткових точок обходу. За підсумком результат передається в модуль відтворення, де додатково вирішується завдання ОЗК. Надали буде детальний розгляд кожного завдання.

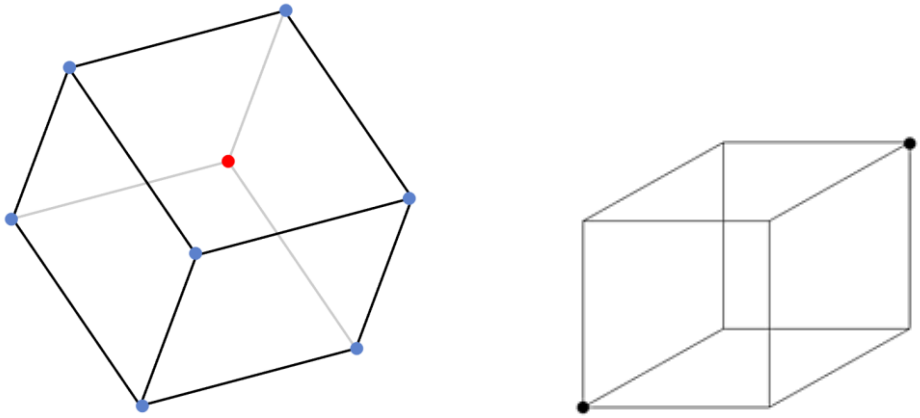
### 3.1.2 Алгоритм знаходження оптимального шляху

Виходячи з результатів в розділі 2.2, було обрано модифікований метод навігаційних графів, який дозволяє оптимально вирішити завдання з пошуку маршруту в 3-х мірному просторі.

Для спрощеного рішення задачі змінимо метод опису обходу об'єктів. У базовій реалізації перешкоди описуються через графи з вершинами і ребрами (рис. 3.2, а), що створює складність в подальшій угрупованню перешкод. Тому перейдемо до опису перешкод у вигляді 2 точок (рис. 3.1, б):

– P min – мінімальна точка щодо початку координат;

– P max – максимальна точка щодо початку координат.

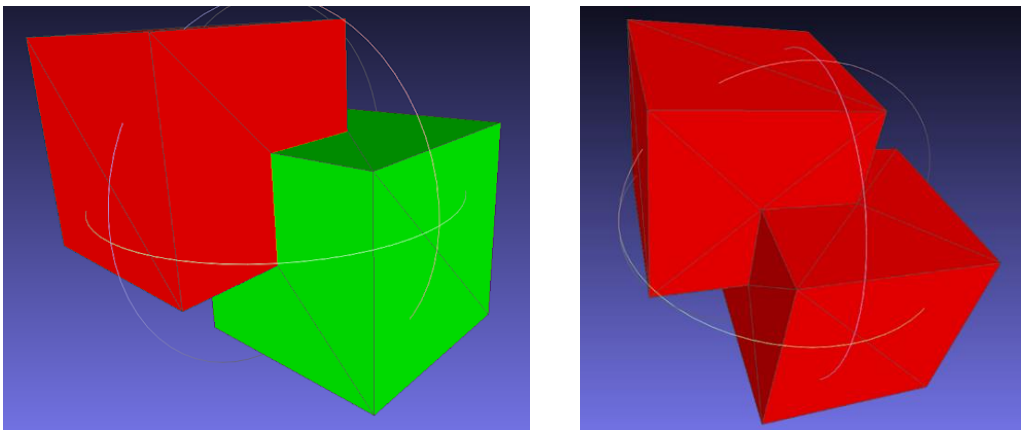


а) б)

Рисунок 3.1 – Bounding boxes

Даний підхід погіршує гнучкість опису перешкод, але в той же час спрощує модель опису. Тобто зараз обхідні «рамки» об'єктів будуть представлені тільки в вигляді прямокутників будь-якого розміру.

Також це спрощення дозволяє легше поєднувати (групувати) перешкоди в просторі. Розглянемо приклад об'єднання перешкод за допомогою графів. В даному прикладі є 2 куба (рис. 3.2, а), графи яких перетинаються. Після комплексних обчислень виходить об'єднаний граф (рис 3.2, б).



а) б)

Рисунок 3.2 – Об'єднання перешкод (графи)

У той же час при об'єднанні перешкод з використання точок P min і P max, нам достатньо знайти нові точки P min і P max (рис. 3.3).

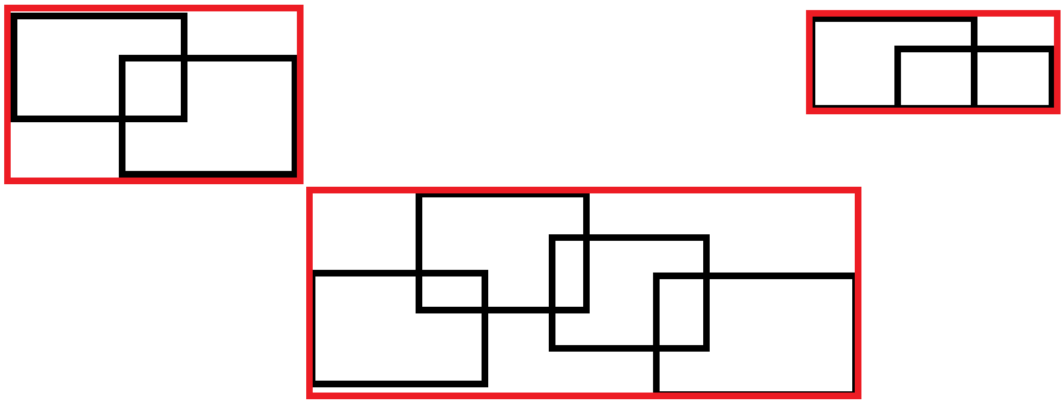


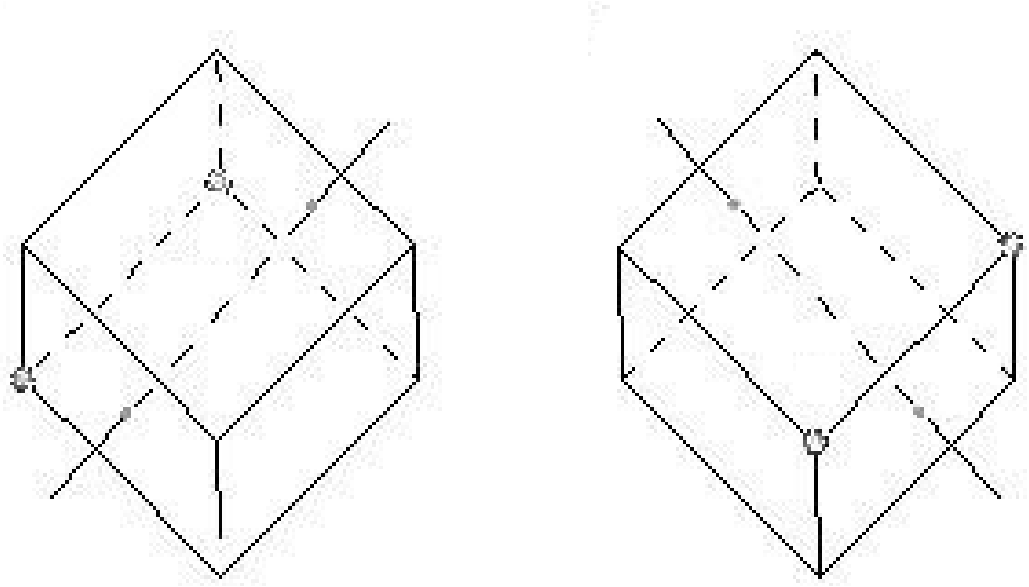
Рисунок 3.3 – Об'єднання перешкод

Наступним етапом знаходимо точки перетину (ТП) з об'єднаними перешкодами між початковою і кінцевою точкою маршруту, що простіше зробити при описі перешкод точками (P max і P min). І сортуємо їх по віддаленості від початкової точки маршруту.

Обхід перешкод за допомогою графів обчислюється через мінімальну суму всіх ребр. У нашому випадку необхідно написати власну логіку обходу перешкод. Так як в даному випадку перешкоди будуть завжди в формі прямокутників, то варіантів обходу вийде 2 типу:

– точки перетину на сусідніх сторонах (рис 3.4, а);

– точки перетину на протилежних сторонах (рис. 3.4, б).



а) б)

Рисунок 3.4 – Приклади обходу

У першому випадку з точками перетину на протилежних сторонах необхідно обчислити 2 додаткові точки для обходу. В інших випадках достатньо 1 додаткової точки.

Надалі вибудовуємо точки для пересування і отримуємо підсумковий маршрут. Рішення завдання пошуку оптимального маршруту завершено і тепер необхідно вирішити зворотну задачу кінематики для маніпулятора.

### 3.1.3 Алгоритм ОЗК

У розділі 2.2 описано метод Девіда-Хантерберга, для вирішення ОЗК з 7 ланками (для 6 зчленувань) на прикладі маніпулятора PUMA. Спростимо модель маніпулятора до 3 зчленувань і вирішимо ОЗК через трапецію (рис. 3.5).

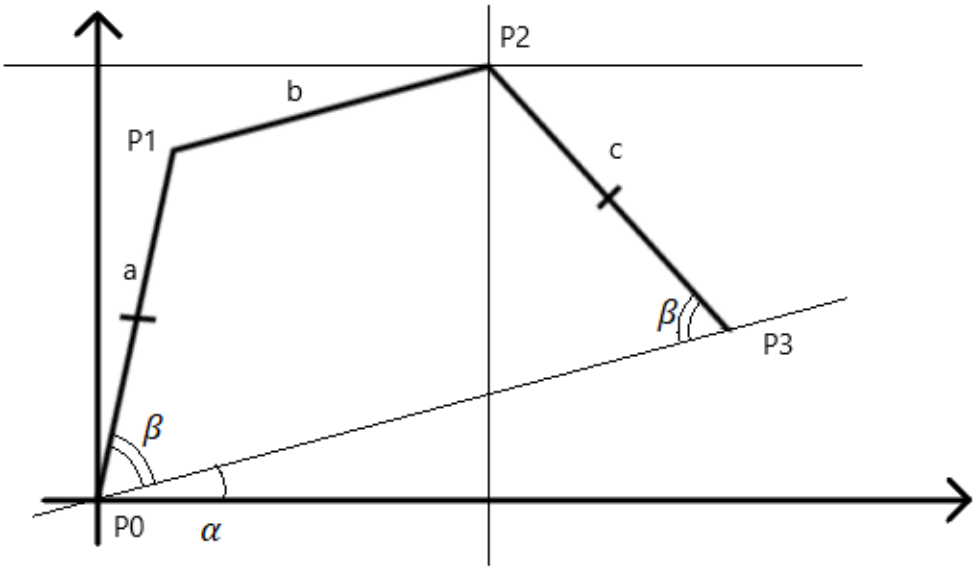



Рисунок 3.5 – Рішення ОЗК через трапецію

Для усунення невизначеностей додаємо умову: бічні сторони завжди будуть рівні. $P_0$ – основа, а $P_3$ – кінцева точка, де знаходиться схват. Кут повороту кожної ланки, виходячи з кінематичної схеми:

– кут повороту ланки «a» – це сума повороту кута трапеції ($\alpha$) і нижнього лівого кута ($\beta$) самої трапеції: $\alpha + \beta$;

– ланка «b» буде паралельна основі трапеції, а значить кут повороту буде дорівнює куту повороту трапеції: $\alpha$;

– ланку «c» можна знайти через трикутники і буде дорівнювати: $\alpha - \beta$.

## 3.2 Розробка програмного забезпечення

### 3.2.1 Загальні відомості

Програмне забезпечення (ПЗ) розроблювалося в IDE Visual Studio на платформі .NET Core з використанням мови програмування C# та бібліотеки OpenTK (Open Toolkit) , яка дозволяє використовувати OpenGL. У додатку А наведено код програми, який частково буде розглянуто далі.

Visual Studio – виконується тільки в Windows. Містить великі вбудовані функції, призначені для роботи з .NET. Випуск Community Edition надається безкоштовно для учнів, учасників проектів з відкритим кодом і окремих користувачів [28].

.NET – це безкоштовна платформа розробки з відкритим вихідним кодом для створення різних типів додатків. Є можливість створювати додатки .NET для багатьох операційних систем [28].

C# – сучасна об'єктно-орієнтована і типобезпечна мова програмування. C # відноситься до широко відомому сімейству мов C, і буде знайома кожному, хто працював з C, C ++, Java або JavaScript [28].

Open Toolkit – це набір швидких низькорівневих прив'язок C # для OpenGL, OpenGL ES і OpenAL. Він працює на всіх основних платформах і підтримує сотні додатків, ігор і наукових досліджень [29].

OpenTK надає кілька службових бібліотек, включаючи пакет математичної та лінійної алгебри, систему управління вікнами і обробку введення [29].

OpenGL (Open Graphics Library) – це специфікація, яка розробляється і підтримується Khronos Group. Має великий набір функцій, що використовуються для управління графікою і зображеннями. Самі функції в основному розробляються виробниками відеокарт.

### 3.2.2 Створення та налаштування проекту

Проект створений за допомогою IDE Visual Studio як консольний додаток .Net Core. Таким чинам буде підтримка декількох операційних систем. Приклад створення зображений на рис.3.6. Але спочатку повинна бути завантажена необхідна версія .Net Core.

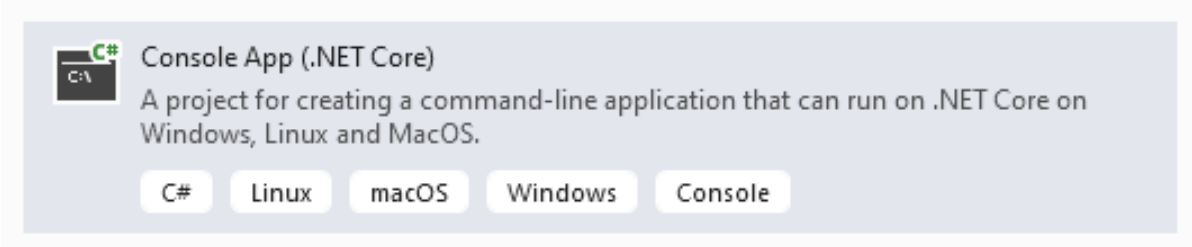



Рисунок 3.6 – Створення консольного додатку

Де в конфігураціях змінений тип виведення на «Windows Application», так як управління відбуватиметься через графічний інтерфейс за допомогою бібліотеки OpenTK (рис. 3.7).

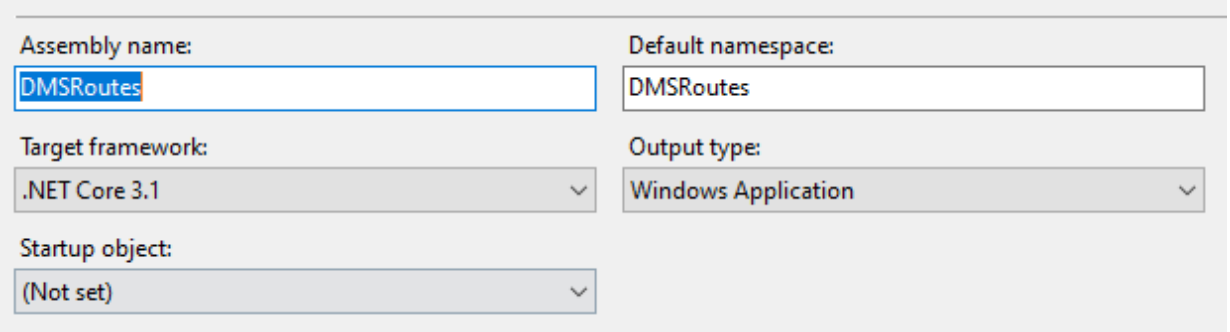



Рисунок 3.7 – Конфігурації проекту

А також в проект додані додаткові модулі (для алгоритмів і відтворення графіки) як бібліотеки класів (рис. 3.8).

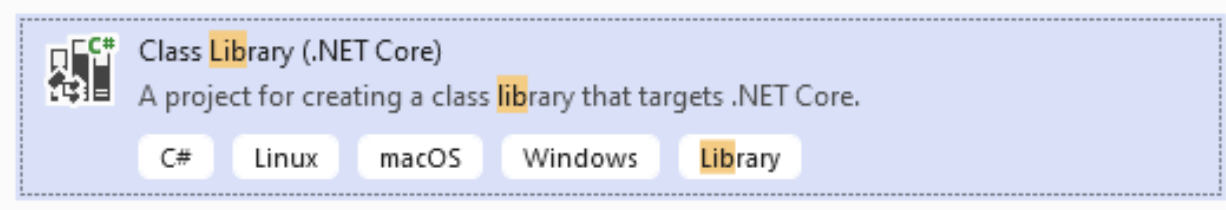



Рисунок 3.8 – Створення додаткових модулів

Через NuGet Package Manager необхідно підключити дві додаткові бібліотеки (рис 3.9):

– OpenTK підключається для графічного модуля;

– Rhino3dm використовується для математичних обчислень і підключається в модулі з алгоритмами.

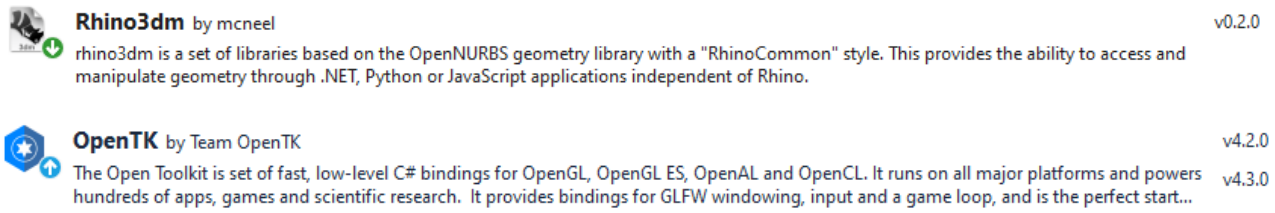



Рисунок 3.9 – Бібліотеки

### 3.2.3 Структура проекту

Сам проект RouteDMS (route decision making system) створений як консольний додаток для збору вхідних даних, запуску алгоритмів прорахунку оптимального маршруту, а також для відтворення простору (рис. 3.10). Тут знаходиться карта простору.

Модуль Algorithms включає в себе набір методів (угруповання перешкод, пошук точок перетину з перешкодами і пошук точок обходу перешкод) для пошуку оптимального шляху модифікованим методом навігаційних графів.

Модуль Drawing безпосередньо взаємодіє з бібліотекою OpenTK для відтворення оточення. У цьому модулі присутні графічні об'єкти для відтворення, шейдери для відео карти, загальні класи для управління камерою, текстурами, шейдерами. А також головний клас для управління всім вікном.

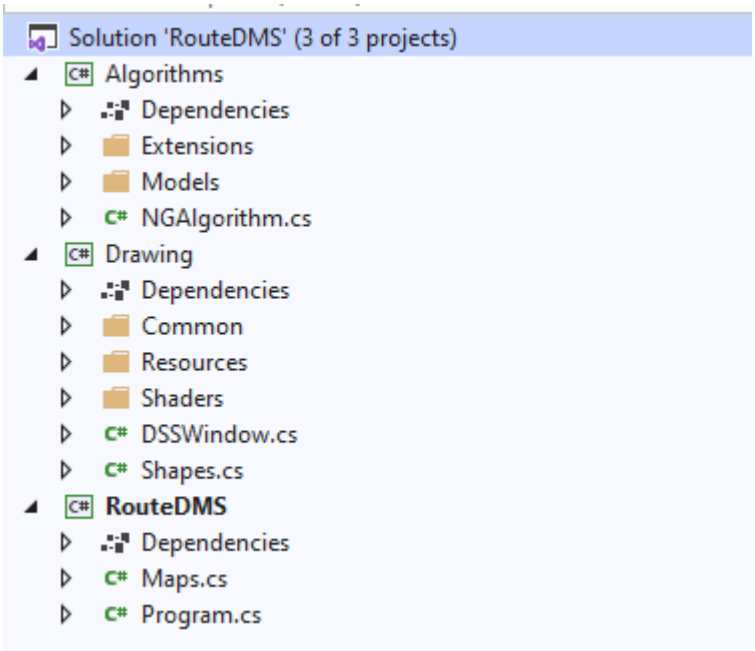



Рисунок 3.10 – Структура проекту

Алгоритм роботи програми представлений на рис. 3.11. На початку створюється об'єкт для відтворення простору DMSWindow, який конфігурується такими параметрами:

– розміри вікна;

– назва вікна;

– частота оновлення кадрів і частота оновлення станів.

Наступним етапом завантажується карта всіх об'єктів. Зараз вона представлена у вигляді масиву координат в статичному класі Maps.

Створюється екземпляр класу NGAlgorithm, який представляє алгоритм для методу НГ і передається в нього карта об'єктів. Далі для вирішення завдання з пошуком маршруту достатньо викликати метод GetRoute, передавши в нього початкову і кінцеву координату шляху. На виході виходить масив точок, які представляють оптимальний маршрут для об'єкта в заданому просторі.

Знайдений маршрут і карту об'єктів необхідно передати в графічний модуль (Drawing), а саме в екземпляр класу DMSWindow. Робиться це за допомогою методів:

– SetMap – для передачі карти;

– SetPoints – для передачі точок маршруту.

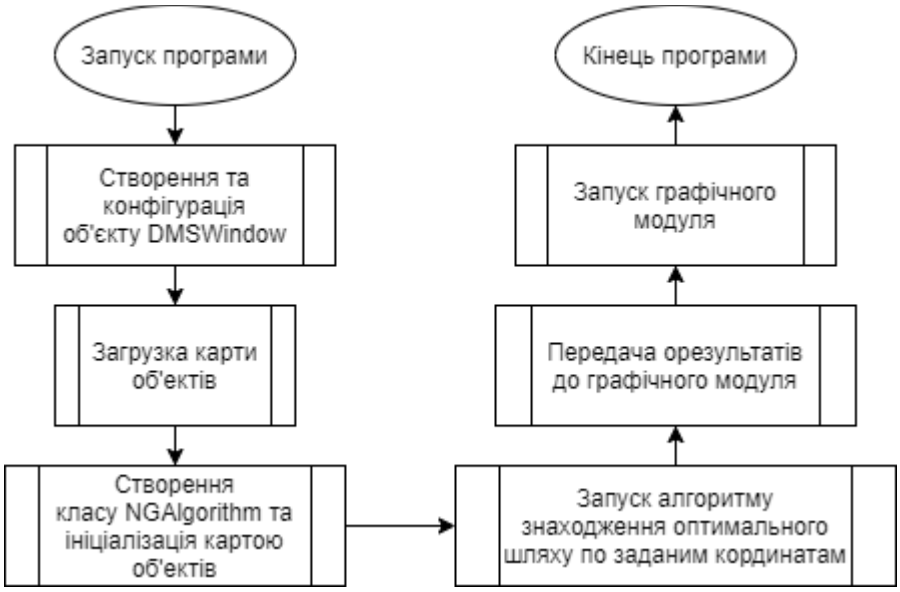



Рисунок 3.11 – Алгоритм роботи програми

Після чого запускається графічний модуль і відбувається прорисовка об'єктів. При кожному запуску оновлення стану відбувається переміщення об'єкта. В цей же час і вирішується ОЗК.

Вікно програми зображено на рис. 3.12. Присутня можливість переміщення камери в просторі. У програмі присутні такі об'єкти:

– перешкоди в формі ящиків (задаються в статичному класі Maps);

– площину, як підставу;

– синя колона (підстава МР);

– зелений куб (схват МР);

– зелені лінії (ланки МР у вигляді кінематичної схеми).

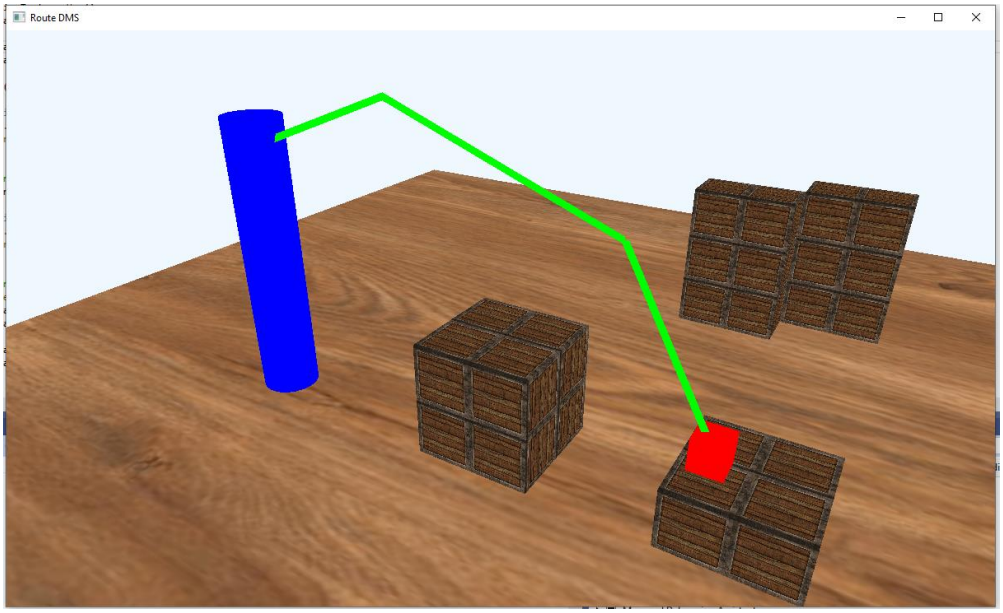



Рисунок 3.12 – Вікно програми

### 3.2.4 Опис функцій проекту

Один з етапів методу навігаційних графів – об'єднання перешкод. Займається цим функція calcIntersectionBoxes. Йде перебір всіх вільних блоків, які ще не брали участі в об'єднанні і не визначені як самостійні, і при виявленні двох сусідніх блоків – створюється один великий, далі цей «великий» блок починає «поглинати» все сусідні блоки. Далі цикл повторюється для вільних блоків (рис. 3.13 – 3.15).

```
private void calcIntersectionBoxes()
{
   IList<BoundingBox> intersected = new List<BoundingBox>();
   foreach (var box in this.initBoxes)
   {
      if (intersected.Contains(box)) continue;
      var intersectedBox= this.searchIntersectionBox(box, intersected);
      if (intersectedBox.HasValue)
      {
         this.unionIntersectionBoxes(box, intersectedBox.Value, intersected);
      } else
      {
         this.usingBoxes.Add(box);
      }
   }
}
```

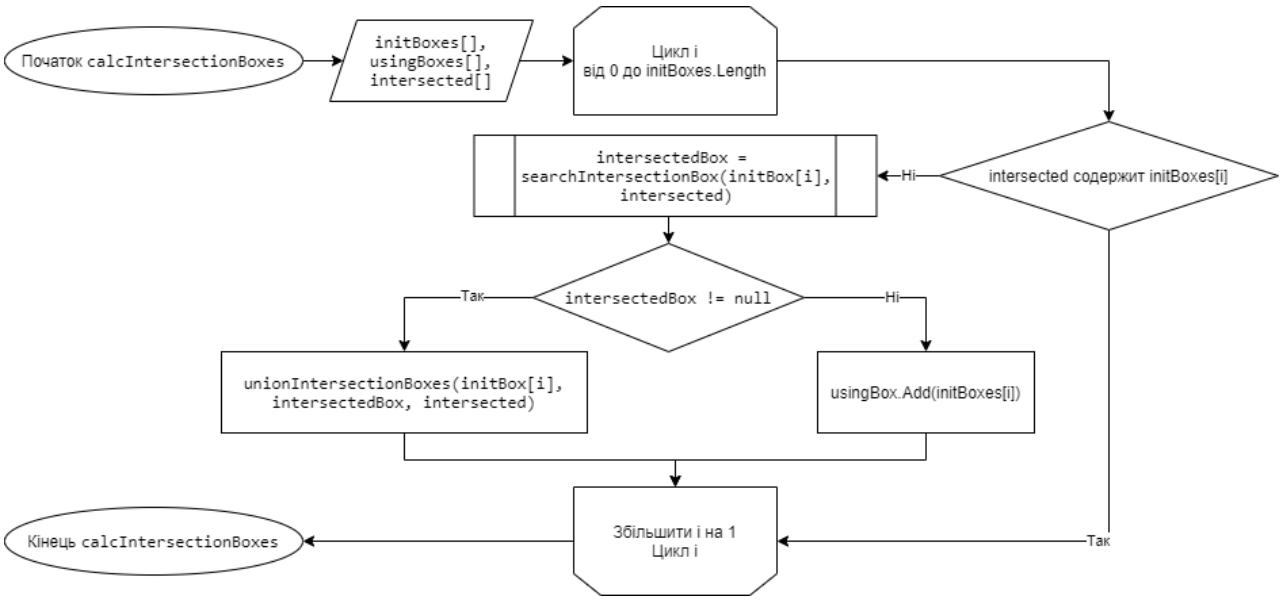



Рисунок 3.13 – Головний алгоритм об'єднання перешкод

```
private BoundingBox? searchIntersectionBox(BoundingBox box, IList<BoundingBox> skip)
{
   foreach (var compareBox in this.initBoxes.Except(skip).Except(this.usingBoxes))
   {
      if (box.Equals(compareBox)) continue;

      if (box.Contains(compareBox.Min) || box.Contains(compareBox.Max))
      {
         return compareBox;
      }
   }

   return null;
}
```

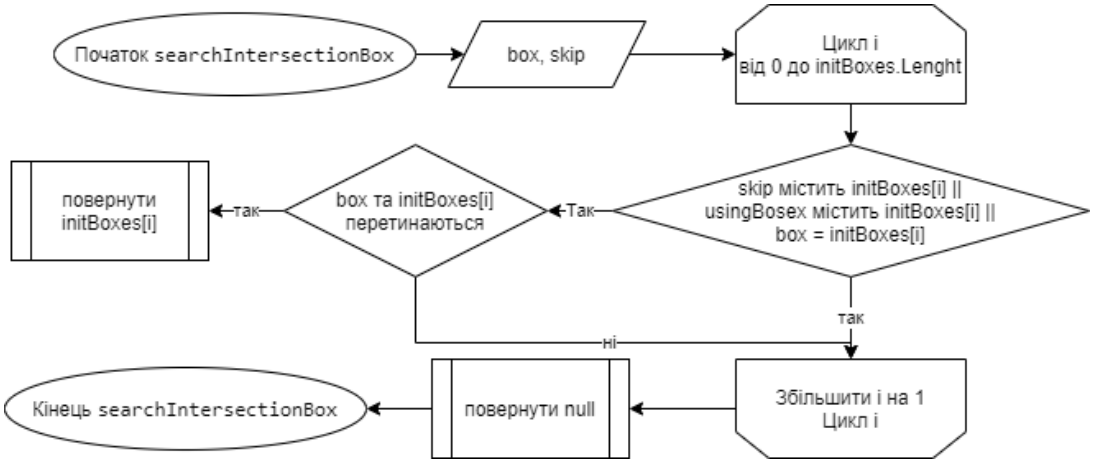



Рисунок 3.14 – Алгоритм пошуку пересічних об’єктів

```
private void unionIntersectionBoxes(BoundingBox box, BoundingBox intersectedBox,
IList<BoundingBox> intersected)
{
   intersected.Add(box);
   intersected.Add(intersectedBox);

   BoundingBox usingBox = BoundingBox.Union(box, intersectedBox);

   bool restart;
   do
   {
      restart = false;
      var newIntersectedBox = this.searchIntersectionBox(usingBox, intersected);
      if (newIntersectedBox.HasValue)
      {
         intersected.Add(newIntersectedBox.Value);
         usingBox.Union(newIntersectedBox.Value);
         restart = true;
      }
   } while (restart);

   this.usingBoxes.Add(usingBox);
}
```

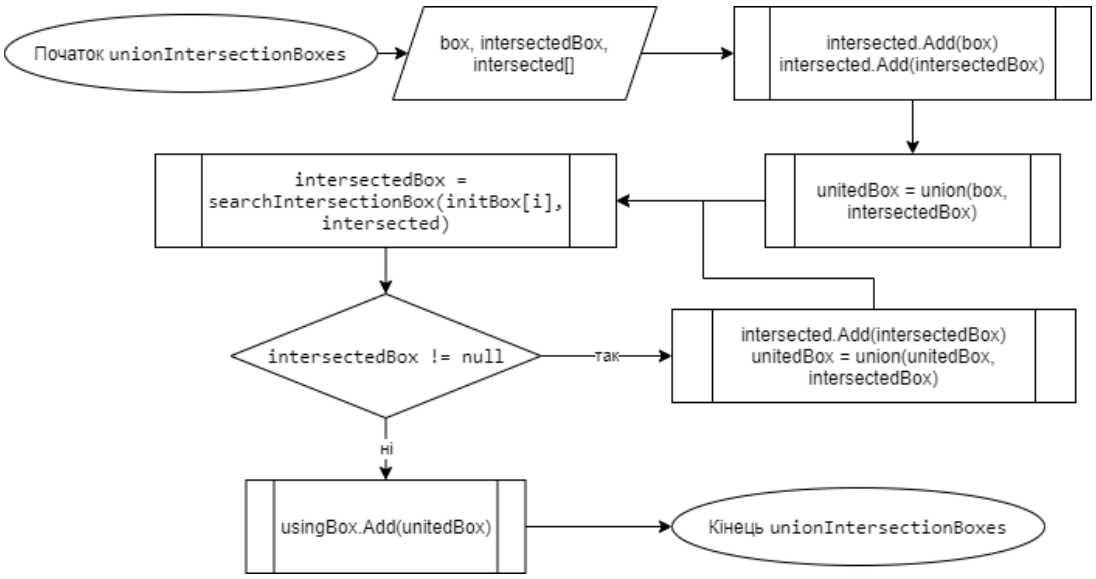



Рисунок 3.15 – Алгоритм об'єднання пересічних об’єктів

Ще одним важливим етапом в методі НГ – це пошук точок перетину (ТП) з перешкодами і точок обходу перешкод (рис. 3.16). Цим займається метод searchIntersectionPoints, який приймаємо пряму – найкоротший маршрут між початком і кінцем маршруту. Спочатку перебираються всі об'єкти в просторі і перевіряються на перетин прямої. І в разі, якщо пряма перетинає цей об'єкт –

починається пошук точок обходу. У нашому випадку якщо будуть перетинатися сусідні сторони – буде одна додаткова точка обходу. Якщо це паралельні сторони – дві точки обходу.

```
private IList<Vector3> searchIntersectionPoints(Line line)
{
   List<Vector3> points = new List<Vector3>();
   var st = line.From.ToVector3();
   points.Add(st);
   foreach (var box in this.usingBoxes)
   {
      if (Intersection.LineBox(line, box, 0, out var res) && res.Min >= 0 && res.Min <= 1 &&
res.Max >= 0 && res.Max <= 1)
      {
         var point1 = line.PointAt(res.Min);
         var point2 = line.PointAt(res.Max);
         points.Add(point1.ToVector3());
         // search additional points
         Point3d[] ps = this.getAdditionalPoints(box, point1, point2);
         points.AddRange(ps.Select(p => p.ToVector3()));
         points.Add(point2.ToVector3());
      }
   }
   var end = line.To.ToVector3();
   points.Add(end);
   return points;
}
```

```
private Point3d[] getAdditionalPoints(BoundingBox bbox, Point3d point1, Point3d point2)
{
   Pos s1 = this.checkSide(bbox.Min, bbox.Max, point1);
   Pos s2 = this.checkSide(bbox.Min, bbox.Max, point2);
   if (this.isParallel(s1, s2))
   {
      var p1 = point1;
      var p2 = point2;
      // calculate p1 and p2
      //
      // ...
      //

      return new Point3d[] { p1, p2 };
   } else
   {
      var p = Point3d.Origin;

      // calculate p
      //
      // ...
      //

      return new Point3d[] { p };
   }
}
```

```
private bool isParallel(Pos s1, Pos s2)
{
   if ((s1 == (s1 & Pos.X) && s2 == (s2 & Pos.X)) ||
       (s1 == (s1 & Pos.Y) && s2 == (s2 & Pos.Y)) ||
       (s1 == (s1 & Pos.Z) && s2 == (s2 & Pos.Z)))
   {
      return true;
   }
   return false;
}

private Pos checkSide(Point3d min, Point3d max, Point3d point)
{
   Pos pos = 0;
   if (point.X == min.X) pos = Pos.Left | pos;
   if (point.X == max.X) pos = Pos.Right | pos;
   if (point.Y == min.Y) pos = Pos.Bottom | pos;
   if (point.Y == max.Y) pos = Pos.Top | pos;
   if (point.Z == min.Z) pos = Pos.Front | pos;
   if (point.Z == max.Z) pos = Pos.Back | pos;

   return pos;
}
```

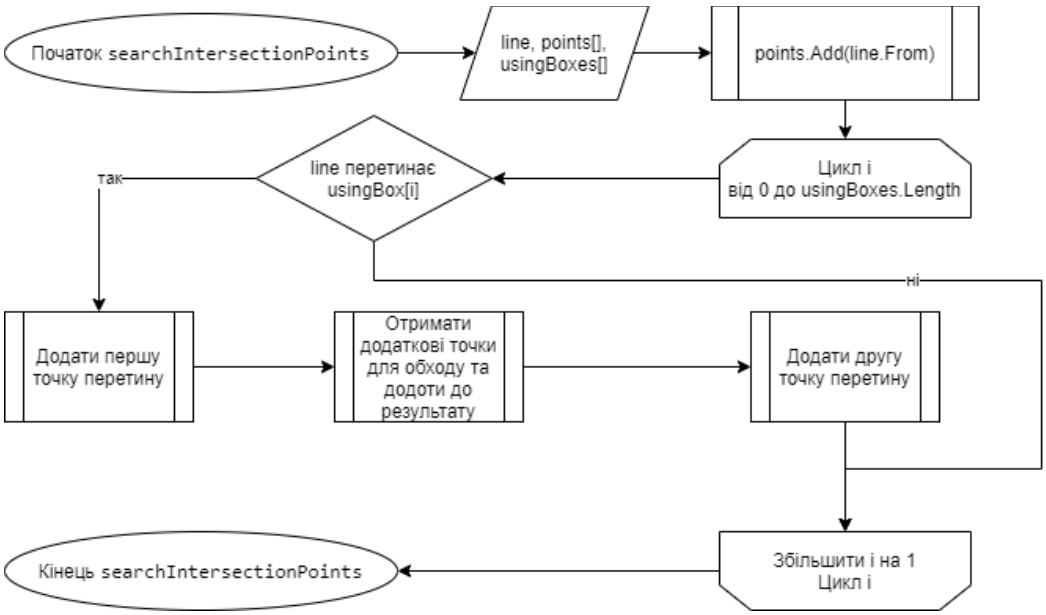



Рисунок 3.16 – Алгоритм пошуку точок перетину і точок обходу

Після знаходження точок маршруту і передачі їх до графічного модулю – приступаємо до вирішення ОЗК. Метод calcIKP обчислює необхідні кути кожного зчленування під час руху руки маніпулятора. Тобто даний метод буде викликатися кожен раз при зміні стану маніпулятора.

```
private void calcIKP()
{
   Vector4 curPos = new Vector4(this.routePoints[this.currentPoint], 1)
                       * Matrix4.CreateTranslation(this.offset * this.direction);

   this.rotateY = MathF.Atan(curPos.Z / curPos.X);

   Vector3 curPL = new Vector3(curPos.X, curPos.Y - this.heightMP, curPos.Z);

   var len = (curPL - Vector3.Zero).Length;
   var A = Vector3.CalculateAngle(new Vector3(curPL.X, 0, curPL.Z), curPL);
   A *= (curPL.Y < 0) ? -1 : 1;
   var B = MathF.Acos((len - this.len_z2) / 2 / this.len_z13);

   this.rotateZ1 = A + B;
   this.rotateZ2 = A;
   this.rotateZ3 = A - B;
}
```

Як вже говорилося раніше, для відтворення простору використовується OpenGL, а саме бібліотеку OpenTK. Тому основне завдання це успадковуватися від класу GameWindow і перевантажити методи OnLoad, OnRenderFrame і OnUpdateFrame.

У методі OnLoad завантажуються текстури, шейдери, вершини об'єктів для відтворення. В методі OnUpdateFrame зазвичай пишеться бізнес-логіка. У нашому випадку це управління камерою, пересування, рішення ОЗК і розрахунок стану маніпулятора. А вже в методі OnUpdateFrame відбувається прорисовка всіх об'єктів, де попередньо обчислюються матриці трансформації об'єктів. Приклад для відтворення ланок маніпулятора:

```
// Drawing line
GL.BindVertexArray(this.lineVAO);
this.defaultShader.SetVector3("objectColor", new Vector3(0.0f, 1.0f, 0.0f));
GL.LineWidth(10);
{
   Matrix4 model = Matrix4.CreateScale(this.len_z13, 0, 0)
               * Matrix4.CreateRotationZ(this.rotateZ1)
               * Matrix4.CreateTranslation(0, this.heightMP, 0)
               * Matrix4.CreateRotationY(this.rotateY);
   this.defaultShader.SetMatrix("model", model);
   GL.DrawArrays(PrimitiveType.Lines, 0, 2);
}
{
   Matrix4 model = Matrix4.CreateScale(this.len_z2, 0, 0)
               * Matrix4.CreateRotationZ(this.rotateZ2 + MathHelper.PiOver2 - this.rotateZ1)
               * Matrix4.CreateTranslation(0, this.len_z13, 0)
```

```
                * Matrix4.CreateRotationZ((MathHelper.PiOver2 - this.rotateZ1) * -1)
                * Matrix4.CreateTranslation(0, this.heightMP, 0)
                * Matrix4.CreateRotationY(this.rotateY);
    this.defaultShader.SetMatrix("model", model);
    GL.DrawArrays(PrimitiveType.Lines, 0, 2);
}
{
    Matrix4 model = Matrix4.CreateScale(this.len_z13, 0, 0)
                * Matrix4.CreateRotationZ(this.rotateZ3 - this.rotateZ2)
                * Matrix4.CreateTranslation(this.len_z2, 0, 0)
                * Matrix4.CreateRotationZ(this.rotateZ2 + MathHelper.PiOver2 - this.rotateZ1)
                * Matrix4.CreateTranslation(0, this.len_z13, 0)
                * Matrix4.CreateRotationZ((MathHelper.PiOver2 - this.rotateZ1) * -1)
                * Matrix4.CreateTranslation(0, this.heightMP, 0)
                * Matrix4.CreateRotationY(this.rotateY); ; //
    this.defaultShader.SetMatrix("model", model);
    GL.DrawArrays(PrimitiveType.Lines, 0, 2);
}
```

## 3.3 Тестування розробленої програми

### 3.3.1 Загальна інформація

Тестування проводилося на персональному комп'ютері (ПК) під операційною системою Windows 10. Характеристики ПК наведені в таблиці 3.1. Було проведено 3 види експерименту з різним оточенням по 3 рази. У кожному експерименті враховувалися такі характеристики і параметри:

– кількість об'єктів в просторі;

– кількість об'єктів на маршруті;

– витрачений час на рішення пошуку маршруту;

– кількість знайдених точок маршруту.

Таблиця 3.1 – Характеристики ПК

| Тип компонента | Модель компонента | Характеристики |
|---|---|---|
| CPU | AMD Ryzen 5 3600X | Тактова частота, ГГц: 3,8 (4,4 Turbo)<br>Кількість ядер: 6<br>Кількість потоків: 12<br>Виробнича технологія, нм: 7 |
| GPU | GeForce GTX 1650 SUPER | Об'єм пам'яті, ГБ: 4<br>Частоти роботи GPU, МГц: 1815<br>Частоти роботи пам'яті, МГц: 12002 |
| RAM | TEAM T-Force Vulcan Z | Обсяг, ГБ: 16<br>Кількість планок в комплекті: 2<br>Тип: DDR4 Стандарт: PC4-25600<br>Ефективна частота, МГц: 3200<br>Штатні таймінги: CL16-18-18-38 |

### 3.3.2 Перший експеримент

У першому експерименті на карті було 24 об'єкти, після об'єднання. вийшло 3 групи перешкод. Завдання полягало в пересуванні схвата з точки (2, 0, 2) через куб (4x4x4) на платформу (4x4x1) в точку (8, 1, 0) (рис. 3.17). В результаті був побудований маршрут з 8 точок (табл. 3.3). Кожна точка представлена як координата у 3-х мірній декартовій системі. Витрачений час на пошук маршруту наведено в таблиці 3.2.

Таблиця 3.2 – Час пошуку маршруту №1

| № спроби | 1 | 2 | 3 |
|---|---|---|---|
| Час пошуку, мс | 61 | 61 | 58 |

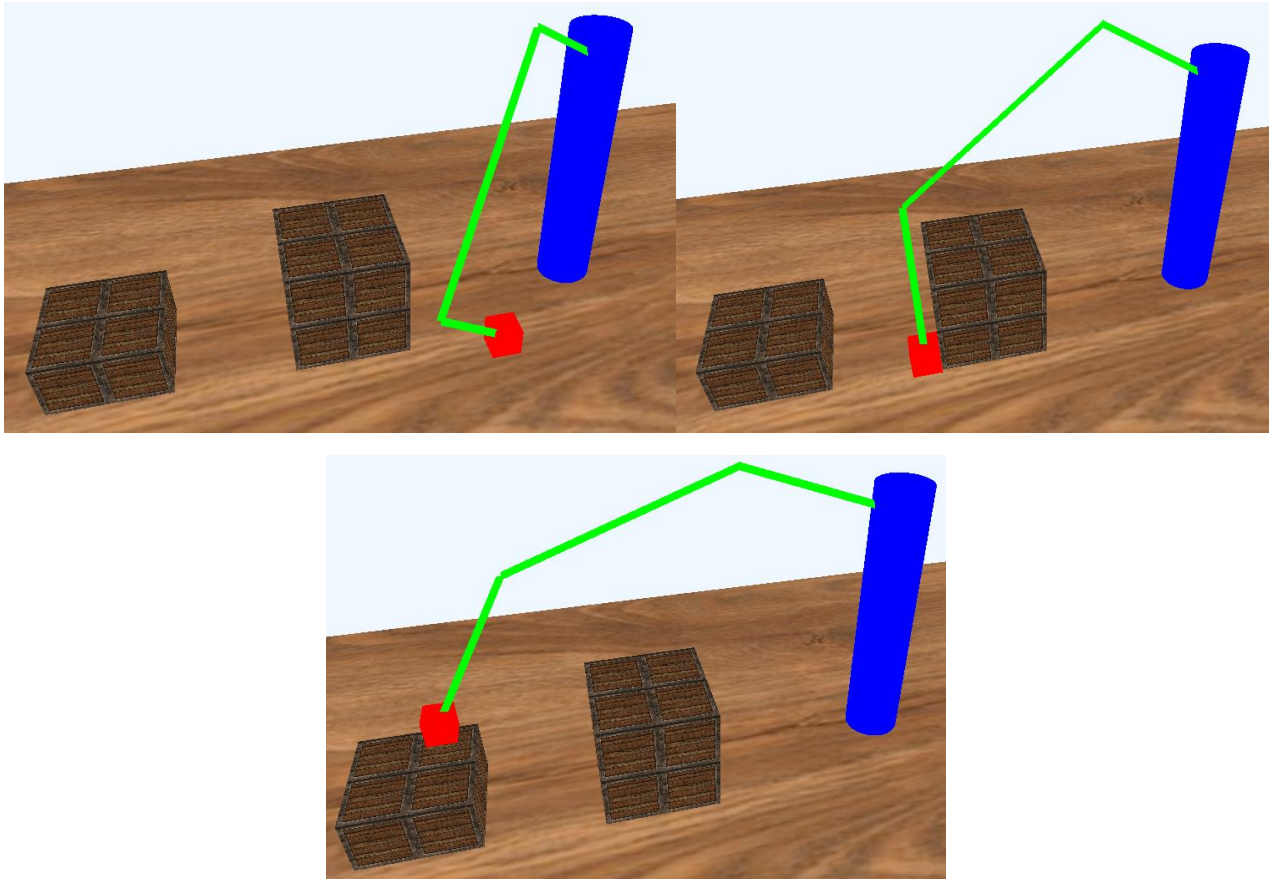


Рисунок 3.17 – Експеримент №1

Таблиця 3.3 – Координати найденого маршруту №1

| № точки | Координата (x, y, z) | № точки | Координата (x, y, z) |
|---|---|---|---|
| 1 | (2, 0, 2) | 5 | (6, 0.67, 0.67) |
| 2 | (3, 0.17, 1.67) | 6 | (7, 0.83, 0.33) |
| 3 | (3, 0.34, 2) | 7 | (7, 1, 0.165) |
| 4 | (6, 0.5, 2) | 8 | (8, 1, 0) |

### 3.3.3 Другий експеримент

У другому експерименті на карті було вже 28 об'єктів, після об'єднання. вийшло також 3 групи перешкод. Завдання полягало в пересуванні схвата з точки (-2, 0, 5) через куб (4x6x4) в точку (3, 0, 4) (рис. 3.18). В результаті був

побудований маршрут з 6 точок (координат) (табл. 3.5). Витрачений час на пошук маршруту наведено в таблиці 3.4.

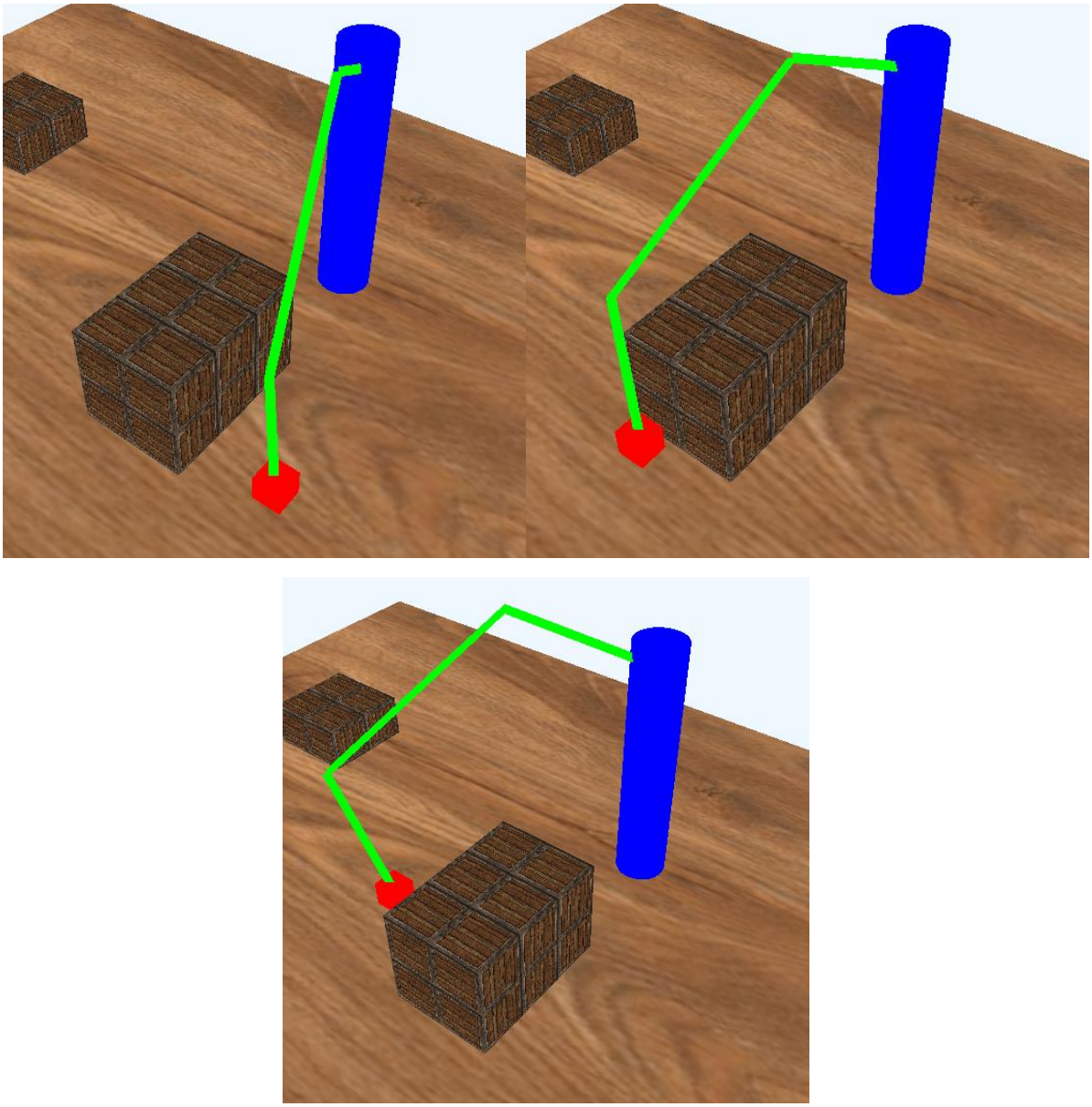


Рисунок 3.18 – Експеримент №2

Таблиця 3.4 – Час пошуку маршруту №2

| № спроби | 1 | 2 | 3 |
|---|---|---|---|
| Час пошуку, мс | 61 | 62 | 65 |

Таблиця 3.5 – Координати найденого маршруту №2

| № точки | Координата (x, y, z) | № точки | Координата (x, y, z) |
|---|---|---|---|
| 1 | (-2, 0, 5) | 4 | (2, 0, 6) |
| 2 | (-1, 0, 4.8) | 5 | (2, 0, 4.2) |
| 3 | (-1, 0, 6) | 6 | (3, 0, 4) |

### 3.3.4 Третій експеримент

У третьому експерименті на карті було теж 28 об'єктів, Нопосле об'єднання. вийшло 5 груп перешкод. Завдання полягало в пересуванні схвата з точки (-5, 0, 2) через 3 перешкоди в точку (4, 0, 4) (рис. 3.19). В результаті був побудований маршрут з 13 точок (координат) (табл. 3.7). Витрачений час на пошук маршруту наведено в таблиці 3.6.

Таблиця 3.6 – Час пошуку маршруту №3

| № спроби | 1 | 2 | 3 |
|---|---|---|---|
| Час пошуку, мс | 60 | 59 | 60 |

Таблиця 3.7 – Координати найденого маршруту №3

| № точки | Координата (x, y, z) | № точки | Координата (x, y, z) |
|---|---|---|---|
| 1 | (-5, 0, 2) | 8 | (1, 0, 4) |
| 2 | (-4, 0, 2.22) | 9 | (1, 0, 3.33) |
| 3 | (-4, 1, 2.37) | 10 | (2, 0, 3.56) |
| 4 | (-2, 1, 2.52) | 11 | (2, 1, 3.7) |
| 5 | (-2, 0, 2.67) | 12 | (4, 1, 3.85) |
| 6 | (-1, 0, 2.89) | 13 | (4, 0, 4) |
| 7 | (-1, 0, 4) | | |

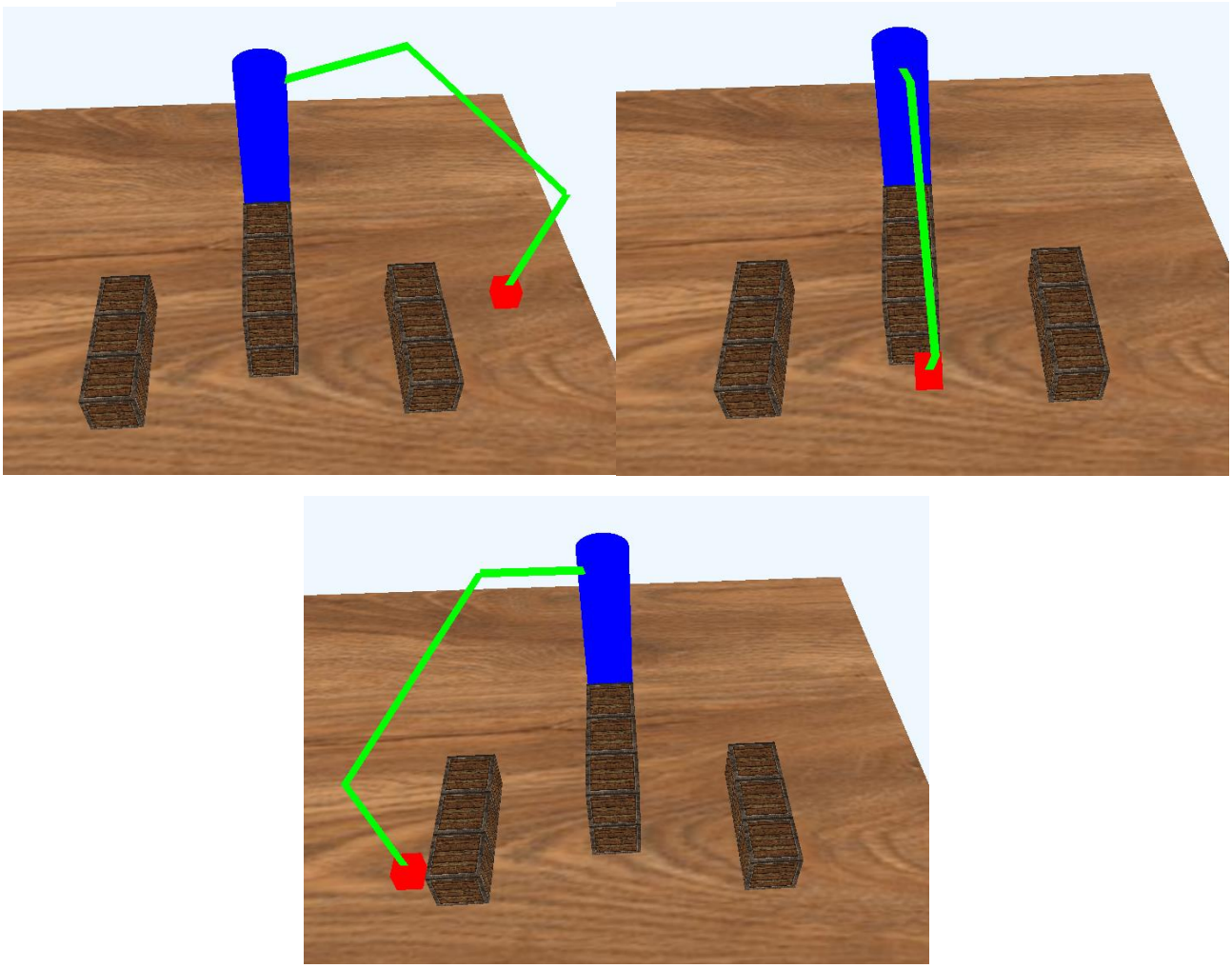


Рисунок 3.19 – Експеримент №3

### 3.3.5 Результати тестування

Проаналізувавши отримані результати маршрутів і провівши візуальне тестування знайдених маршрутів в розробленому ПЗ, можна зробити висновок, що обрані та розроблені методи і алгоритми підходять для вирішення завдань з пошуком оптимального маршруту. Витрачений час на пошук маршруту займає десятки мілісекунд, що дозволяє використовувати інтелектуальну систему прийняття рішень в реальному часі і вести перерахунок маршруту при кожному зіткненні з динамічними об'єктами в просторі. У таблиці 3.8 представлені підсумкові результати трьох експериментів.

Таблиця 3.8 – Підсумкові результати тестування

| номер експерименту | кількість об'єктів | Кількість об'єктів, що перетинаються | Витрачений час на пошук, мс | Кількість координат в маршруті |
|---|---|---|---|---|
| 1 | 24 | 2 | 60 | 8 |
| 2 | 28 | 1 | 62 | 6 |
| 3 | 28 | 3 | 59,6 | 13 |

## 3.4 Техніка безпеки та розрахунок освітленості під час роботи за комп'ютером

У багатьох населених пунктах напруга в мережі може сильно коливатися. Для комп'ютера такі зміни напруги є небажаними, тому краще підключати комп'ютери через стабілізатори. Найбільш надійний захист від неприємностей, пов'язаних з нестабільністю електроживлення, здійснюють спеціальні пристрої безперервного живлення (UPS – Uninterruptable Power Supply), які не тільки забезпечують строго постійну напругу, але і дають можливість роботи комп'ютерів при повному відключенні електроживлення протягом від 5 хвилин до декількох годин. За цей час можна повністю завершити роботи, які ведуться на комп'ютері, щоб при виключенні не відбулося втрати інформації. Для серверів локальних мереж і комп'ютерів, що обробляють цінну інформацію застосування пристроїв безперервного живлення є практично обов'язковим.

Перед початком експлуатації комп'ютера слід перевірити, чи відповідає напруга в мережі тій, на яку розрахований комп'ютер.

Системний блок комп'ютера бажано поставити в такому місці, щоб він не піддавався поштовхам і вібраціям. Всі кабелі, що з'єднують системний блок

комп'ютера з іншими пристроями, слід вставляти та виймати тільки при вимкненому комп'ютері

В якості профілактичних заходів для забезпечення пожежної безпеки слід використовувати скриту електромережу, надійні розетки з пожежобезпечних матеріалів, силові мережі живлення виконувати кабелями, розрахованими на підключення навантаження в 3...5 разів більше від використовуємого, включати й виключати живлення обладнання за допомогою штатних вимикачів. Треба регулярно робити очистку внутрішніх частин комп'ютерів та іншого устаткування від пилу, розташовувати комп'ютери на окремих столах. Для запобігання іскріння необхідно рідше встромляти і виймати штепсельні вилки з розеток.

Метою світлотехнічного розрахунку є розробка рекомендацій щодо розташування оптимальної кількості світильників в приміщенні для створення комфортних, які відповідають всім нормам умов перебування людини.

Одним з найбільш важливих якісних показників висвітлення, що регламентуються нормативними документами, є коефіцієнт пульсації. Для офісних приміщень нормований коефіцієнт пульсації відповідно до ДСанПіН 5.5.6.009-98 становить не більше 5%. Найбільш простим і ефективним способом усунення пульсацій світлового потоку є використання світильників з електронної пускорегулювальної апаратурою. Проведемо розрахунки в разі використання світильників PRBLUX 436.

За методом коефіцієнтів використання необхідна кількість світильників N в освітлювальної установки визначається за допомогою формули:

$$N = \frac{E_{Н} \times S \times K_{З}}{K_{И} \times n \times \Phi_{л}} \qquad (3.1)$$

де $E_Н$ – нормативний рівень освітленості, лк;

S – площа приміщення, м$^2$;

КЗ – коефіцієнт запасу для освітлювальних установок загального освітлення за ДСанПіН 5.5.6.009-98 дорівнює 5,4;

Ки – коефіцієнт використання;

n – кількість ламп в світильнику;

$Ф_л$ – світловий потік однієї лампи в світильнику.

Основним критерієм, за яким визначається необхідна кількість освітлювальних приладів, є нормований рівень освітленості ЕН. Цей показник для приміщення за ДСанПіН 5.5.6.009-98 становить 40 лк для розрахункової площині на висоті 0,8 м від підлоги.

Площу приміщення визначимо за формулою:

$$S = a \times b,$$

де S – площа приміщення, м$^2$;

a – довжина приміщення, м;

b – ширина приміщення, м.

Розрахуємо площу приміщення:

$$S = 8 \times 7 = 56\ (\text{м}^2).$$

Значення коефіцієнтів відбиття наведені в таблиці 3.9.

Таблиця 3.9 – Коефіцієнти відбиття

| Поверхня, що відбиває | Коефіцієнт відбиття |
|---|---|
| площу з білою поверхнею (побілена стеля; побілені стіни з вікнами, закриті білими шторами) | 70 |
| площу зі світлою поверхнею (побілені стіни при вікнах без штор; побілена стеля в сирих приміщеннях; чиста бетонна і світла дерев'яна стеля) | 50 |
| площу з сірою поверхнею (бетонну стелю в брудних приміщеннях; дерев'яна стеля; бетонні стіни з вікнами, стіни обклеєні світлими шпалерами) | 30 |
| площу з темною поверхнею (стіни і стелі в приміщеннях з великою кількістю темної пилу; суцільне скління без штор; червона неоштукатурена цегла, стіни з темними шпалерами) | 10 |

Виходячи з таблиці 3.9 і з того, що стіни і стеля обклеєні світлими шпалерами, а підлога вкрита темним ламінатом коефіцієнти відбиття в приміщені для стін, стелі і підлоги будуть 30, 30 і 10 відповідно.

Коефіцієнт використання у світильника PRBLUX 436 дорівнює 0,5.

Кількість ламп в світильнику обраного типу становить 5, кожна з яких має світловий потік $Ф_Л$ = 1200 лм.

Знайдемо необхідну кількість світильників, скориставшись виразом (3.1):

$$N = \frac{400 \times 56 \times 1{,}4}{0{,}5 \times 5 \times 1200} = 10.45$$

Таким чином, для даного приміщення освітлювальна установка повинна складатися з 11 обраних світильників з рівномірним розподілом по поверхні стелі.

### 3.5 Висновки

Використовуючи модифікований метод навігаційних графів для пошуку оптимального шляху було розроблено ПЗ та проведено тестування. Розроблене рішення є ефективне та швидкодіюче, працює у реальному часі. Може бути використано в інтелектуальній системі прийняття рішень.

# ВИСНОВКИ

Проаналізувавши конструктивні характеристики маніпуляційних промислових роботів і підходи до розробки інтелектуальних системи прийняття рішень МР можна виділити кілька головних напрямків при створенні таких систем:

– пошук оптимальних маршрутів в 3-вимірному просторі, причому характеристика оптимальності може залежати від конкретного завдання і умов на виробництві;

– рішення задач кінематики;

– облік динаміки маніпулятора.

На якість переміщення маніпулятора впливають динамічні сили: відцентрові, коріолісові і т. д., які необхідно враховувати в реальному світі. При моделюванні їх можна опустити.

Задачі кінематики діляться на два типи: пряма і зворотна. При вирішенні прямої задачі кінематики шукається позиція схвата МР, знаючи кінематичну схему МР та його стан. При вирішенні ОЗК шукається стан ланок при відомій позиції схвата МР і його кінематичної схеми. У розділі 2.2 наводиться приклад рішення за допомогою методу Девіда-Хантерберга на прикладі робота PUMA. При моделюванні системи в нашому випадку було необхідно тільки рішення ОЗК. Також було спрощено завдання і вирішено через трапецію.

Аналіз відомих методів для пошуку маршруту показав, що найефективніший був модифікований метод навігаційного графа. Цей метод ефективно вирішує завдання пошуку маршруту в 3-х мірному просторі. При цьому за допомогою даного метод легко враховуються динамічні об'єкти в просторі будь-якої форми і в будь-якій кількості. Також модифікований метод дозволив позбутися трудомісткої роботи для опису навігаційних графів для кожного об'єкта.

В результаті було розроблено ПЗ для моделювання СУ МР, в якому застосовуються обрані методи з пошуку оптимального маршруту і рішення ОЗК. Тестування показало, що дані методи ефективні і можуть застосовуватися в реальному часі. За підсумком була отримана інтелектуальна система прийняття рішень, яка має ряд переваг:

– швидкодію;

– адекватність побудованого маршруту;

– облік динамічних об'єктів в будь-якій кількості і будь-якої форми.

Розроблене алгоритмічне і програмне забезпечення може бути складовою частиною інтелектуальної системи прийняття рішень для окремого маніпуляційного робота і для гнучкої інтегрованої роботизованої системи.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення [Текст]. – Введ. 2015-06-22. – К.: Держстандарт України, 2017. – 29 с.

2. Невлюдов, І.Ш. Дипломне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» [Текст]: навч. посіб. / І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, Г.В. Пономарьова. – Київ-58, пр. Космонавта Комарова, 1, 2019. – 320 с.

3. Методичні вказівки з «Розробки й оформлення магістерської атестаційної роботи» для студентів другого (магістерського) рівня вищої освіти галузі знань 15 Автоматизація та приладобудування за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані технології освітні програми: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І.Ш. Невлюдов, В.В. Косенко, В.В. Євсєєв. – Харків: ХНУРЕ, 2019. – 55 с.

4. Дієсперов А., Цимбал О. Вибір середовища візуалізації процесів інтелектуальної системи прийняття рішень для аналізу якості рішень // Матеріали 4-ї Міжнародної конференції Виробництво & Мехатронні Системи. 22-23 жовтня 2020 р. – Харків: ХНУРЕ, 2020. – С. 63 – 65.

5. Дієсперов А., Цимбал О. Вибір середовища візуалізації процесів інтелектуальної системи прийняття рішень для аналізу якості рішень // Збірник наукових статей «ADED-2020» (Випуск 2) – Харків : ХНУРЕ, 2020. – С. 27 – 31.

6. ДЕСТ 25686-85. Маніпулятори, автооператори і промислові роботи. М .: Видавництво стандартів, 1985.

7. Фу К., Гонсалес Р., Лі К. Робототехніка. – М.: Мир – 1989. – 624 с.

8. Юревич Є.І. Управління роботами і робототехнічними системами. СПб. – 2001. – 168 с.

9. ElMaraghy Hoda. Changable and Reconfigurable Manufacturing Systems. – Springler, 2009. – 405 p.

10. ДЕСТ 25685-85. Роботи промислові. Класифікація. – М.: Видавництво стандартів, 1983.

11. ДЕСТ 12.2.072-82. Роботи промислові, роботизовані технологічні комплекси і ділянки. Загальні вимоги безпеки. – М.: Видавництво стандартів, 1982.

12. А.Н. Горитщв, С.М. Алфьоров. Згладжування траєкторій переміщення робочого органу робота маніпулятора // Вість Томського політехнічного університету. – Томськ, 2006 № 8. – с. 176 – 179.

13. Фролов К.В. Механіка промислових роботів. Кн. 1: Кінематика і динаміка. – М.: Вища школа – 1988. – 304 с.

14. Черноусько Ф.Л., Болотник М.М., Градецький В.Г. Маніпуляційні роботи: динаміка, управління, оптимізація. М.: Наука, 1989. – 368 с

15. ДЕСТ 4.480-87. Систем показників якості продукції робототехніки, номенклатура основних показників. – М.: Видавництво стандартів, 1988.

16. Лебедєв П.А. Кінематика просторових механізмів. – М.: Машинобудування, 1967. – 280 с.

17. R. S. Hartenberg and J. Denavit, «A kinematic notation for lower pair mechanisms based on matrices» Journal of Applied Mechanics, vol. 77, pp. 215–221, June 1955.

18. Tolani D., Goswami A., Badler N. I. Real-time inverse kinematics techniques for anthropomorphic limbs // Graphical models. 2000. Vol. 62, no. 5. P. 353–388.

19. Тертичний-Даури В. Динаміка робототехнічних систем. Санкт-Петербург, 2012. – 128 с.

20. Віттенбурге Й. Динаміка систем твердих тіл. – М.: Мир, 1980 – 292 с.

21. R. Featherstone, D. Orin, Robot Dynamics: Equations and Algorithms, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000.

22. I. Nevliudov, O. Tsymbal, A. Andrusevitch, V. Gopejenko. Intelligent Decision-Making Support for Flexible Integrated manufacturing – Riga: ISMA, 2020. – 390 p.

23. Роботизовані виробничі комплекси / Ю. Г. Козирєв, А. А. Кудінов, В. Е. Булатов та ін .; Під ред. Ю. Г. Козирєва, А. А. Кудінова. – М.: Машинобудування, 1987. – 270 с.

24. Artem Bronnikov, Nevliudov Igor, Oleksandr Tsymbal. Flexible manufacturing tendencies and improvements with visual sensoring / Eskisehir Technical University Journal of Science and Technology. Applied Sciences and Engineering, 2019. Vol. 20, ICONAT issue, P. 77-83.

25. Дієсперов А., Цимбал О. Використання OpenMP в системах керування мобільними роботами // Матеріали 23-го Міжнародного форуму Радіоелектроніка та молодь у XXI столітті. 16-18 квітня 2019 р. – Харків: ХНУРЕ, 2020. – С. 61 – 62.

26. Peter Yap, “Grid-Based Path-Finding,” AI '02 Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence.

27. Armstrong B., Khatib O., Burdick J. The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm // Proceedings. 1986 IEEE International Conference on Robotics and Automation. 1986.

28. Введення в .NET [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/.

29. OpenTK [Електронний ресурс] – Режим доступу до ресурсу: https://opentk.net/.